

Ottimizzazione: come la Natura risolve i problemi

INTRODUZIONE GENERALE

La natura è un'instancabile risoltrice di problemi. Ogni struttura che osserviamo nel mondo naturale, dalla delicata architettura di una ragnatela alla complessa geometria di un alveare, rappresenta una soluzione ottimale a vincoli specifici. Quando una bolla di sapone assume la sua forma sferica perfetta, sta risolvendo un problema matematico profondo: minimizzare l'area superficiale per un dato volume. Quando una formica traccia il percorso verso una fonte di cibo, sta partecipando a un processo collettivo di ottimizzazione del percorso. Quando le foglie di un girasole si dispongono secondo una spirale matematicamente precisa, stanno massimizzando l'esposizione alla luce solare disponibile.

Questi fenomeni non sono casuali né misteriosi. Sono governati da principi matematici fondamentali che possiamo descrivere, analizzare e applicare. Il calcolo variazionale, sviluppato da matematici come Johann Bernoulli, Leonhard Euler e Joseph-Louis Lagrange nei secoli diciassettesimo e diciottesimo, fornisce gli strumenti per comprendere come la natura trova queste soluzioni ottimali. Le equazioni che emergono da questa teoria non sono semplici descrizioni astratte, ma rappresentano le leggi fisiche fondamentali che governano il comportamento dei sistemi naturali.

Questo trattato esplora la profonda connessione tra matematica e natura attraverso il prisma dell'ottimizzazione. Partiremo dai fondamenti teorici del calcolo variazionale, esaminando i problemi classici che hanno dato origine a questa disciplina. Procederemo poi ad analizzare come questi principi si manifestano nei sistemi biologici e fisici, dalla geometria degli alveari alla struttura delle reti vascolari. Nella terza parte, vedremo come gli informatici hanno tratto ispirazione da questi processi naturali per creare algoritmi potenti e versatili. Infine, esploreremo le applicazioni moderne di questi concetti, dalla logistica all'apprendimento automatico.

La bellezza di questo approccio risiede nella sua universalità. Gli stessi principi matematici che descrivono la forma di una catenaria appesa tra due punti spiegano anche la struttura ottimale di un ponte sospeso. Le equazioni che governano la disposizione delle foglie su un fusto vegetale trovano applicazione nella progettazione di antenne e pannelli solari. La natura, attraverso miliardi di anni di evoluzione, ha scoperto soluzioni che noi stiamo ora imparando a comprendere e replicare.

PARTE I: CALCOLO VARIAZIONALE

Capitolo 1: Fondamenti Matematici del Calcolo Variazionale

1.1 Il Concetto di Funzionale

Prima di affrontare i problemi classici del calcolo variazionale, dobbiamo comprendere il concetto fondamentale su cui si basa questa disciplina: il funzionale. Nella matematica ordinaria, una funzione associa un numero a un altro numero. Per esempio, la funzione $f(x) = x^2$ associa a ogni numero reale x il suo quadrato. Un funzionale, invece, è un oggetto matematico più astratto: associa un numero a un'intera funzione.

Consideriamo un esempio concreto. Supponiamo di voler calcolare la lunghezza di una curva nel piano che connette due punti fissati. Se la curva è descritta dalla funzione $y(x)$, dove x varia da x_0 a x_1 , la lunghezza totale è data dall'integrale:

$$L[y] = \int_{(x_0 \text{ a } x_1)} \sqrt{1 + (dy/dx)^2} dx$$

Qui L è un funzionale: prende in input un'intera funzione $y(x)$ e restituisce un singolo numero, la lunghezza della curva. Questo è radicalmente diverso dal calcolo ordinario, dove valutiamo funzioni in punti specifici. Nel calcolo variazionale, stiamo valutando proprietà globali di funzioni intere.

Il problema centrale del calcolo variazionale è questo: data una famiglia di funzioni che soddisfano certi vincoli (per esempio, tutte le curve che connettono due punti dati), quale funzione rende minimo (o massimo) un certo funzionale? Questo è esattamente il tipo di domanda che la natura "risolve" continuamente. Una corda appesa tra due punti assume la forma che minimizza la sua energia potenziale. Un raggio di luce viaggia lungo il percorso che minimizza il tempo di percorrenza.

1.2 Il Problema della Brachistocrona

Il problema della brachistocrona, proposto da Johann Bernoulli nel 1696, rappresenta uno dei problemi fondanti del calcolo variazionale. La parola "brachistocrona" deriva dal greco e significa "tempo più breve". Il problema è il seguente: data una particella che si muove sotto l'influenza della gravità, senza attrito, quale curva dovrebbe seguire per muoversi da un punto A a un punto B più in basso nel minor tempo possibile?

A prima vista, si potrebbe pensare che la risposta sia una linea retta, essendo questa la distanza più breve. Tuttavia, la fisica del problema è più sottile. Una particella che scende lungo una curva più ripida all'inizio acquisterà velocità più rapidamente, e questa velocità maggiore può compensare una distanza totale maggiore. Il problema diventa quindi trovare il compromesso ottimale tra distanza e velocità.

Formuliamo il problema matematicamente. Poniamo il punto di partenza A nell'origine del sistema di coordinate, con l'asse y che punta verso il basso. Il punto di arrivo B si trova in posizione (x_1, y_1) . Una particella che parte da fermo in A e si muove lungo una curva $y(x)$ ha, in un punto generico (x, y) , una velocità determinata dalla conservazione dell'energia:

$$\frac{1}{2}mv^2 = mgy$$

dove m è la massa della particella, v la sua velocità e g l'accelerazione gravitazionale. Da questa relazione otteniamo:

$$v = \sqrt{2gy}$$

L'elemento di lunghezza lungo la curva è $ds = \sqrt{1 + (dy/dx)^2} dx$. Il tempo impiegato per percorrere un elemento infinitesimo di curva è $dt = ds/v$. Il tempo totale è quindi:

$$T[y] = \int_0^{x_1} \frac{\sqrt{1 + (dy/dx)^2}}{\sqrt{2gy}} dx$$

Questo è il funzionale che vogliamo minimizzare. La soluzione a questo problema, scoperta da Bernoulli e altri matematici dell'epoca, è sorprendentemente elegante: la curva ottimale è una cicloide, la curva tracciata da un punto su un cerchio che rotola lungo una linea retta.

La cicloide può essere descritta parametricamente come:

$$x(\theta) = r(\theta - \sin(\theta)) \quad y(\theta) = r(1 - \cos(\theta))$$

dove r è il raggio del cerchio generatore e θ è il parametro che varia da 0 fino a un valore θ_1 determinato dalle coordinate del punto finale B. Questa curva ha proprietà notevoli: non solo minimizza il tempo di discesa, ma è anche tautocrona, il che significa che il tempo di discesa è indipendente dal punto di partenza sulla curva.

La scoperta della soluzione alla brachistocrona non fu solo un trionfo matematico, ma dimostrò il potere del nuovo approccio variazionale. Johann Bernoulli, Leibniz, Newton, e il fratello di Johann, Jacob Bernoulli, tutti trovarono la soluzione utilizzando metodi diversi, ma convergenti. Questo problema segnò l'inizio di una nuova era nella matematica e nella fisica.

1.3 La Catenaria: La Curva della Corda Appesa

Un altro problema classico che illustra magnificamente i principi del calcolo variazionale è quello della catenaria. Quando una corda o una catena flessibile viene appesa tra due punti fissi, assume una forma particolare sotto l'influenza della gravità. Questa curva, chiamata catenaria (dal latino "catena"), non è una parabola come si potrebbe ingenuamente supporre, ma ha una forma matematica specifica che emerge dalla minimizzazione dell'energia potenziale.

Consideriamo una corda uniforme di densità lineare ρ (massa per unità di lunghezza) appesa tra due punti. Ogni elemento infinitesimo di corda di lunghezza ds ha massa $dm = \rho ds$ e si trova a un'altezza y rispetto a un livello di riferimento. La sua energia potenziale gravitazionale è $dU = \rho g y ds$, dove g è l'accelerazione gravitazionale.

L'energia potenziale totale della corda è:

$$U[y] = \rho g \int y \sqrt{1 + (dy/dx)^2} dx$$

La corda assume la configurazione che minimizza questa energia, soggetta al vincolo che la sua lunghezza totale è fissata. Questo è un problema variazionale con vincolo, che può essere risolto usando il metodo dei moltiplicatori di Lagrange nel contesto variazionale.

La soluzione a questo problema è la catenaria, descritta dall'equazione:

$$y(x) = a \cosh(x/a) = a(e^{(x/a)} + e^{(-x/a)})/2$$

dove a è una costante che dipende dalla lunghezza della corda e dalla distanza tra i punti di sospensione, e $\cosh$ è la funzione coseno iperbolico. Questa curva ha proprietà geometriche affascinanti. La tensione in ogni punto della corda è proporzionale alla curvatura in quel punto, e la forma della catenaria è tale da distribuire uniformemente le forze lungo tutta la sua lunghezza.

La catenaria non è solo un esercizio matematico astratto. Quando viene ruotata attorno al suo asse verticale, genera una superficie chiamata catenoide, che è una superficie minimale (ne parleremo più avanti). Inoltre, se una catenaria viene capovolta, la sua forma diventa quella di un arco ideale che sopporta il proprio peso senza richiedere rinforzi. L'architetto catalano Antoni Gaudí utilizzò modelli fisici di catenarie appese per progettare la struttura della Sagrada Família, capovolgendo i modelli per ottenere archi che sopportassero ottimalmente il peso.

1.4 Superfici Minime e Film di Sapone

Quando immergiamo una struttura metallica a forma di anello in una soluzione di sapone e la estraiamo, si forma un film sottile che attraversa l'anello. Questo film assume spontaneamente una forma che minimizza la sua area superficiale, soggetta al vincolo di essere ancorato al bordo dell'anello. Questo è un esempio fisico perfetto di un problema variazionale in due dimensioni.

La tensione superficiale del film di sapone crea una forza che tende a contrarre la superficie. L'energia del sistema è proporzionale all'area totale della superficie, quindi il film assume la configurazione di area minima. Matematicamente, stiamo cercando una superficie $z(x, y)$ che minimizza il funzionale dell'area:

$$A[z] = \iint \sqrt{1 + (\partial z / \partial x)^2 + (\partial z / \partial y)^2} \, dx \, dy$$

soggetta a condizioni al bordo specificate dalla forma dell'anello.

La condizione necessaria affinché una superficie sia minimale è che la sua curvatura media sia zero in ogni punto. La curvatura media H di una superficie è definita come la media delle due curvature principali (le curvature massima e minima in un punto dato). Per una superficie descritta come grafico di una funzione $z(x, y)$, la condizione $H = 0$ si traduce nell'equazione differenziale:

$$(1 + z_y^2)z_{xx} - 2z_x z_y z_{xy} + (1 + z_x^2)z_{yy} = 0$$

dove z_x indica la derivata parziale rispetto a x , e così via. Questa è l'equazione delle superfici minime, un'equazione differenziale parziale non lineare del secondo ordine.

La superficie minimale più semplice è il piano, ma esistono molte altre superfici minimali interessanti. Il catenoide, che abbiamo menzionato prima, è la superficie di rivoluzione minimale ottenuta ruotando una catenaria attorno al suo asse. L'elicoide, che assomiglia a una scala a chiocciola infinita, è anch'essa una superficie minimale. Nel 1776, Jean Baptiste Meusnier dimostrò che queste erano le uniche due superfici minimali di rivoluzione (oltre al piano).

Un risultato notevole, dimostrato solo negli anni '90 del ventesimo secolo, è che esistono superfici minimali di complessità arbitraria. Il matematico brasiliano Celso Costa scoprì nel 1982 una superficie minimale con la topologia di un toro (una ciambella) con un punto rimosso, dimostrando che le superfici minimali possono avere strutture topologiche molto più ricche di quanto si pensasse.

I film di sapone sono "computer analogici" naturali che risolvono problemi di ottimizzazione. Se creiamo una struttura metallica complicata, il film di sapone che si forma troverà automaticamente la superficie di area minima che connette i bordi. Questo principio è stato utilizzato dagli architetti per progettare strutture a tensione, come le coperture degli stadi, e dai matematici per visualizzare e studiare superfici minimali complesse.

1.5 Le Equazioni di Eulero-Lagrange

Finora abbiamo discusso problemi specifici del calcolo variazionale, ma non abbiamo ancora sviluppato il metodo generale per risolverli. Questo metodo generale fu sviluppato da Leonhard Euler e Joseph-Louis Lagrange nella metà del diciottesimo secolo e porta il loro nome: le equazioni di Eulero-Lagrange.

Consideriamo il problema variazionale generale: trovare una funzione $y(x)$ che minimizza (o massimizza) un funzionale della forma:

$$I[y] = \int_{(x_0 \text{ a } x_1)} F(x, y, y') dx$$

dove $y' = dy/dx$ e F è una funzione data di tre variabili. La funzione $y(x)$ deve passare attraverso due punti fissi: $y(x_0) = y_0$ e $y(x_1) = y_1$.

L'idea chiave è considerare una variazione della funzione $y(x)$. Supponiamo che $y(x)$ sia la soluzione ottimale. Consideriamo una funzione leggermente perturbata:

$$\tilde{y}(x) = y(x) + \varepsilon \eta(x)$$

dove ε è un numero molto piccolo e $\eta(x)$ è una funzione arbitraria che si annulla agli estremi: $\eta(x_0) = \eta(x_1) = 0$. Questa condizione assicura che $\tilde{y}(x)$ passi ancora attraverso i punti fissi.

Il valore del funzionale per la funzione perturbata è:

$$I(\varepsilon) = \int_{(x_0 \text{ a } x_1)} F(x, y + \varepsilon \eta, y' + \varepsilon \eta') dx$$

Se $y(x)$ è effettivamente la funzione che minimizza I , allora $I(\varepsilon)$ deve avere un minimo quando $\varepsilon = 0$. Questo significa che la derivata di $I(\varepsilon)$ rispetto a ε deve annullarsi quando $\varepsilon = 0$:

$$dI/d\varepsilon|_{\varepsilon=0} = 0$$

Calcoliamo questa derivata. Usando la regola della catena:

$$dI/d\varepsilon = \int_{(x_0 \text{ a } x_1)} [\partial F/\partial y \, \eta + \partial F/\partial y' \, \eta'] \, dx$$

Integriamo per parti il secondo termine:

$$\int_{(x_0 \text{ a } x_1)} \partial F/\partial y' \, \eta' \, dx = [\partial F/\partial y' \, \eta]_{(x_0)}^{(x_1)} - \int_{(x_0 \text{ a } x_1)} d/dx(\partial F/\partial y') \, \eta \, dx$$

Il termine di bordo si annulla perché $\eta(x_0) = \eta(x_1) = 0$. Quindi:

$$dI/d\varepsilon|_{\varepsilon=0} = \int_{(x_0 \text{ a } x_1)} [\partial F/\partial y - d/dx(\partial F/\partial y')] \, \eta \, dx = 0$$

Poiché questa equazione deve valere per ogni funzione $\eta(x)$ arbitraria, l'espressione tra parentesi quadre deve essere identicamente nulla. Questo ci dà l'equazione di Eulero-Lagrange:

$$\partial F/\partial y - d/dx(\partial F/\partial y') = 0$$

Questa è l'equazione differenziale che la funzione ottimale $y(x)$ deve soddisfare. È un'equazione del secondo ordine, poiché coinvolge la derivata seconda di y attraverso il termine $d/dx(\partial F/\partial y')$.

Applichiamo questo risultato al problema della brachistocrona. In quel caso, abbiamo:

$$F(x, y, y') = \sqrt{(1 + y'^2)}/\sqrt{(2gy)}$$

Calcoliamo le derivate parziali:

$$\partial F/\partial y = -\sqrt{(1 + y'^2)}/(2y\sqrt{(2gy)}) = -\sqrt{(1 + y'^2)}/(2y)^{3/2}\sqrt{(2g)}$$

$$\partial F/\partial y' = y'/[\sqrt{(2gy)}\sqrt{(1 + y'^2)}]$$

L'equazione di Eulero-Lagrange diventa piuttosto complicata, ma può essere semplificata notando che F non dipende esplicitamente da x . In questo caso, esiste una prima integrazione che dà:

$$F - y' \partial F/\partial y' = \text{costante}$$

Questa relazione, dopo alcuni passaggi algebrici, porta alla soluzione della cicloide che abbiamo menzionato prima.

1.6 Generalizzazioni e Vincoli

Le equazioni di Eulero-Lagrange che abbiamo derivato si applicano al caso più semplice: un funzionale che dipende da una singola funzione di una variabile. Tuttavia, molti problemi interessanti richiedono generalizzazioni.

Funzionali con derivate superiori: Se il funzionale dipende non solo da y e y' , ma anche da derivate superiori come y'' , y''' , ecc., le equazioni di Eulero-Lagrange si generalizzano. Per un funzionale:

$$I[y] = \int F(x, y, y', y'') dx$$

l'equazione di Eulero-Lagrange diventa:

$$\partial F / \partial y - d/dx(\partial F / \partial y') + d^2/dx^2(\partial F / \partial y'') = 0$$

Più funzioni: Se il problema coinvolge più funzioni $y_1(x)$, $y_2(x)$, ..., $y_n(x)$, otteniamo un sistema di n equazioni di Eulero-Lagrange, una per ciascuna funzione.

Più variabili indipendenti: Per superfici o campi in più dimensioni, dove cerchiamo una funzione $u(x_1, x_2, \dots, x_n)$, l'equazione di Eulero-Lagrange diventa un'equazione differenziale parziale. Per esempio, per un funzionale:

$$I[u] = \iint F(x, y, u, u_x, u_y) dx dy$$

l'equazione è:

$$\partial F / \partial u - \partial / \partial x(\partial F / \partial u_x) - \partial / \partial y(\partial F / \partial u_y) = 0$$

Vincoli: Spesso i problemi variazionali includono vincoli. Per esempio, nel problema della catenaria, la lunghezza della corda è fissata. Questi vincoli possono essere trattati usando i moltiplicatori di Lagrange, proprio come nell'ottimizzazione ordinaria.

Un vincolo isoperimetrico è un vincolo sulla forma:

$$\int G(x, y, y') dx = \text{costante}$$

In presenza di tale vincolo, il funzionale modificato da minimizzare diventa:

$$I[y] = \int [F(x, y, y') + \lambda G(x, y, y')] dx$$

dove λ è il moltiplicatore di Lagrange, una costante da determinare insieme alla funzione $y(x)$.

Capitolo 2: Quindici Problemi Classici Risolti

2.1 Il Percorso di Luce Minimo: Principio di Fermat

Pierre de Fermat, nel 1662, enunciò un principio che avrebbe avuto profonde conseguenze per la fisica e la matematica: la luce viaggia tra due punti lungo il percorso che richiede il minor tempo. Questo principio, apparentemente semplice, spiega sia la riflessione che la rifrazione della luce.

Consideriamo prima la riflessione. Un raggio di luce viaggia da un punto A a un punto B, riflettendosi su uno specchio piano. Il percorso totale consiste in due segmenti rettilinei: da A

al punto di riflessione P sullo specchio, e da P a B. La luce viaggia in linea retta in un mezzo omogeneo (questo stesso deriva dal principio di Fermat), quindi la distanza totale è:

$$L = \sqrt{(x_A - x_P)^2 + y_A^2} + \sqrt{(x_B - x_P)^2 + y_B^2}$$

dove abbiamo posto lo specchio lungo l'asse x, e A si trova in (x_A, y_A) , B in (x_B, y_B) , e P in $(x_P, 0)$. Per minimizzare L rispetto a x_P , deriviamo e poniamo uguale a zero:

$$dL/dx_P = -(x_A - x_P)/\sqrt{(x_A - x_P)^2 + y_A^2} + (x_B - x_P)/\sqrt{(x_B - x_P)^2 + y_B^2} = 0$$

Questa equazione si può riscrivere come:

$$\sin(\theta_i) = \sin(\theta_r)$$

dove θ_i è l'angolo di incidenza (angolo tra il raggio incidente e la normale allo specchio) e θ_r è l'angolo di riflessione. Quindi otteniamo la legge della riflessione: l'angolo di incidenza è uguale all'angolo di riflessione.

La rifrazione è ancora più interessante. Quando la luce passa da un mezzo a un altro (ad esempio dall'aria all'acqua), cambia direzione. Supponiamo che la luce viaggi con velocità v_1 nel primo mezzo e v_2 nel secondo. Il tempo totale di percorrenza da A a B, con cambio di mezzo nel punto P sull'interfaccia, è:

$$T = \sqrt{(x_A - x_P)^2 + y_A^2}/v_1 + \sqrt{(x_B - x_P)^2 + y_B^2}/v_2$$

Minimizzando rispetto a x_P , otteniamo:

$$\sin(\theta_1)/v_1 = \sin(\theta_2)/v_2$$

Poiché l'indice di rifrazione n è definito come $n = c/v$ (dove c è la velocità della luce nel vuoto), questa diventa la legge di Snell:

$$n_1 \sin(\theta_1) = n_2 \sin(\theta_2)$$

Il principio di Fermat è profondo perché mostra che le leggi dell'ottica geometrica possono essere derivate da un singolo principio variazionale. Questo prefigura lo sviluppo della meccanica lagrangiana e hamiltoniana, dove tutta la fisica classica viene formulata in termini di principi variazionali.

2.2 La Geodetica: Il Percorso Più Breve su una Superficie

Una geodetica è la curva di lunghezza minima che connette due punti su una superficie. Su un piano, le geodetiche sono linee rette. Su una sfera, sono archi di cerchi massimi (cerchi il cui centro coincide con il centro della sfera). Ma come troviamo le geodetiche su superfici più complesse?

Consideriamo una superficie parametrizzata da coordinate (u, v) . Un punto sulla superficie ha posizione $r(u, v)$. Una curva sulla superficie può essere descritta parametricamente come $u(t), v(t)$, dove t è un parametro che varia lungo la curva. La lunghezza di questa curva è:

$$L = \int \sqrt{(E(du/dt)^2 + 2F(du/dt)(dv/dt) + G(dv/dt)^2)} dt$$

dove E, F, e G sono i coefficienti della prima forma fondamentale della superficie:

$$E = |\partial r / \partial u|^2 \quad F = (\partial r / \partial u) \cdot (\partial r / \partial v) \quad G = |\partial r / \partial v|^2$$

Applicando le equazioni di Eulero-Lagrange a questo funzionale, otteniamo le equazioni delle geodetiche. Per una sfera di raggio R, usando le coordinate sferiche (θ , φ) dove θ è la colatitudine e φ è la longitudine, abbiamo:

$$E = R^2 \quad F = 0 \quad G = R^2 \sin^2(\theta)$$

Le equazioni delle geodetiche, dopo alcuni calcoli, confermano che i cerchi massimi sono effettivamente i percorsi di lunghezza minima.

Le geodetiche hanno un'importanza enorme in fisica. Nella relatività generale di Einstein, i corpi in caduta libera seguono geodetiche nello spaziotempo curvo. La gravità non è una forza nel senso newtoniano, ma è la manifestazione della curvatura dello spaziotempo, e i pianeti orbitano attorno al Sole seguendo geodetiche in questo spaziotempo curvo.

2.3 Il Problema Isoperimetrico

Il problema isoperimetrico classico chiede: tra tutte le curve chiuse di perimetro dato, quale racchiude l'area massima? La risposta intuitiva è il cerchio, e questo può essere dimostrato rigorosamente usando il calcolo variazionale.

Parametizziamo una curva chiusa nel piano usando coordinate polari centrate in un punto interno alla curva. La curva è descritta da $r(\theta)$ dove θ varia da 0 a 2π . Il perimetro è:

$$L = \int_0^{2\pi} \sqrt{r^2 + (dr/d\theta)^2} d\theta$$

e l'area racchiusa è:

$$A = \frac{1}{2} \int_0^{2\pi} r^2 d\theta$$

Vogliamo massimizzare A soggetti al vincolo che L è costante. Usando un moltiplicatore di Lagrange λ , formiamo il funzionale:

$$I[r] = \int_0^{2\pi} \left[\frac{1}{2} r^2 + \lambda \sqrt{r^2 + (dr/d\theta)^2} \right] d\theta$$

L'equazione di Eulero-Lagrange per questo funzionale è:

$$r - \lambda \frac{d}{d\theta} \left[r' / \sqrt{r^2 + r'^2} \right] = 0$$

dove $r' = dr/d\theta$. Questa equazione differenziale può essere risolta, e la soluzione è:

$$r(\theta) = \text{costante}$$

che descrive un cerchio. Inoltre, si può dimostrare che questo è effettivamente un massimo, non solo un punto stazionario.

Questo risultato spiega perché le bolle di sapone sono sferiche: una sfera è la superficie tridimensionale di area minima che racchiude un volume dato, la generalizzazione tridimensionale del problema isoperimetrico bidimensionale.

2.4 La Trattrice e la Pseudosfera

Immaginiamo di trascinare un oggetto pesante lungo il terreno tirando con una corda di lunghezza fissa. Se inizialmente l'oggetto è fermo e la corda è tesa perpendicolarmente alla direzione di moto, quale traiettoria seguirà l'oggetto?

Questo problema porta alla curva chiamata trattrice. Ponendo che la corda abbia lunghezza a e che tiriamo l'oggetto lungo l'asse x positivo, la curva $y(x)$ seguita dall'oggetto soddisfa la condizione che la porzione di corda dall'oggetto al punto dove la tiriamo ha sempre lunghezza a . Geometricamente, questo significa che la porzione di tangente alla curva tra il punto di tangenza e l'asse x ha lunghezza a .

Questa condizione si traduce nell'equazione differenziale:

$$dy/dx = -y/\sqrt{(a^2 - y^2)}$$

Risolvendo questa equazione (separando le variabili e integrando), otteniamo:

$$x = a \ln[(a + \sqrt{(a^2 - y^2)})/y] - \sqrt{(a^2 - y^2)}$$

Questa curva, la trattrice, ha proprietà geometriche notevoli. Quando viene ruotata attorno al suo asintoto (l'asse x), genera una superficie chiamata pseudosfera, che ha curvatura gaussiana costante negativa. Questa superficie è importante nella geometria non euclidea: rappresenta un modello della geometria iperbolica.

2.5 La Cicloide come Curva Tautocrona

Abbiamo già incontrato la cicloide come soluzione al problema della brachistocrona. Sorprendentemente, la cicloide ha un'altra proprietà notevole: è tautocrona, il che significa che il tempo impiegato da una particella per scivolare sotto gravità dalla sua posizione di partenza al punto più basso è indipendente dalla posizione di partenza.

Consideriamo una cicloide con l'equazione parametrica:

$$x = r(\theta - \sin \theta) \quad y = r(1 - \cos \theta)$$

dove il punto più basso è in $\theta = 0$. Se una particella parte da fermo in un punto corrispondente all'angolo θ_0 , il tempo per raggiungere il fondo può essere calcolato usando la conservazione dell'energia. La velocità in funzione dell'angolo θ è:

$$v = \sqrt{2gr(\cos \theta - \cos \theta_0)}$$

L'elemento di arco lungo la cicloide è:

$$ds = 2r \sin(\theta/2) d\theta$$

Il tempo di discesa è:

$$T = \int_{(\theta_0 \text{ a } 0)} ds/v = \int_{(\theta_0 \text{ a } 0)} 2r \sin(\theta/2) / \sqrt{[2gr(\cos \theta - \cos \theta_0)]} d\theta$$

Usando l'identità trigonometrica $\cos \theta = 1 - 2\sin^2(\theta/2)$ e facendo una sostituzione, si può dimostrare che questo integrale vale:

$$T = \pi\sqrt{r/g}$$

Notevolmente, questo tempo è indipendente da θ_0 ! Ogni particella, indipendentemente da dove parte sulla cicloide, impiega lo stesso tempo per raggiungere il fondo. Questa proprietà fu scoperta da Christiaan Huygens nel 1673 e ha applicazioni pratiche nella progettazione di orologi a pendolo di precisione.

La proprietà tautocrona suggerisce che la cicloide potrebbe essere la soluzione a un altro problema variazionale: trovare la curva lungo la quale il periodo delle oscillazioni di una particella è indipendente dall'ampiezza. Infatti, se invertiamo una cicloide (con il punto più basso in alto), una particella che oscilla in questo "pozzo" cicloidale avrà un periodo costante, risolvendo un problema che affliggeva i pendoli ordinari, il cui periodo dipende dall'ampiezza delle oscillazioni.

2.6 La Curva di Delaunay e le Superfici di Rivoluzione

Quando una superficie minimale viene generata per rotazione di una curva attorno a un asse, la curva generatrice deve soddisfare condizioni speciali. Il matematico francese Charles-Eugène Delaunay studiò sistematicamente questo problema nel 1841.

Consideriamo una curva $r(z)$ nel piano rz (coordinate cilindriche), dove r è la distanza dall'asse z . Quando questa curva viene ruotata attorno all'asse z , genera una superficie di rivoluzione. L'area di questa superficie è:

$$A = 2\pi \int r\sqrt{1 + (dr/dz)^2} dz$$

Per trovare le superfici minimali di rivoluzione, applichiamo l'equazione di Eulero-Lagrange con $F = r\sqrt{1 + r'^2}$, dove $r' = dr/dz$. L'equazione risultante è:

$$r'' = (1 + r'^2)/r$$

dove $r'' = d^2r/dz^2$. Questa equazione può essere integrata una volta per dare:

$$r/\sqrt{1 + r'^2} = c$$

dove c è una costante di integrazione. Risolvendo per r' , otteniamo:

$$dr/dz = \pm\sqrt{(r^2/c^2 - 1)}$$

Questa equazione differenziale ha diverse soluzioni a seconda dei valori di r e c :

1. **Il catenoide:** Se $r \geq c$, la soluzione è: $r(z) = c \cosh(z/c)$

Questa è la catenaria che, ruotata, genera il catenoide, l'unica superficie minimale di rivoluzione non banale oltre al piano.

2. **Il piano:** Se $r = c$ costante, otteniamo un cilindro di raggio c , che però non è minimale. La soluzione minimale degenerata è il piano ($c \rightarrow \infty$, z costante).

Delaunay scoprì che tutte le superfici di rivoluzione a curvatura media costante (non necessariamente zero) possono essere generate dalla rotazione delle curve elastiche, che a loro volta sono le orbite di un pendolo nel piano delle fasi. Questo collega elegantemente la geometria differenziale alla meccanica classica.

2.7 Il Problema di Plateau

Il problema di Plateau, dal nome del fisico belga Joseph Plateau che lo studiò sperimentalmente nel diciannovesimo secolo, chiede: data una curva chiusa nello spazio tridimensionale, esiste una superficie minimale che ha quella curva come bordo? Se sì, è unica?

Plateau osservò sperimentalmente che i film di sapone formano superfici minimali quando vengono tesi su strutture di fil di ferro. Dimostrare matematicamente l'esistenza di tali superfici si rivelò estremamente difficile. La soluzione completa al problema fu ottenuta solo nel 1930 da Jesse Douglas e Tibor Radó, indipendentemente. Douglas ricevette una delle prime medaglie Fields nel 1936 per questo lavoro.

Consideriamo alcuni esempi concreti. Se la curva di bordo è un cerchio, esistono due soluzioni al problema di Plateau: il disco piatto (la soluzione ovvia) e il catenoide (se consideriamo due cerchi paralleli come bordo). Quale superficie si forma dipende dalla distanza tra i cerchi e dal loro raggio.

Matematicamente, il problema può essere formulato così: trovare una funzione $u(x, y)$ definita su un dominio D nel piano xy , tale che:

1. u assume valori specificati sul bordo di D (condizioni di Dirichlet)
2. u minimizza l'area della superficie $z = u(x, y)$

Questo porta all'equazione delle superfici minimali:

$$(1 + u_y^2)u_{xx} - 2u_x u_y u_{xy} + (1 + u_x^2)u_{yy} = 0$$

Questa equazione differenziale parziale non lineare è notoriamente difficile da risolvere. Anche determinare se una soluzione esiste richiede tecniche sofisticate di analisi funzionale.

Un risultato sorprendente è che il problema di Plateau può avere multiple soluzioni. Per esempio, Ennio De Giorgi e altri matematici hanno costruito esempi di curve di bordo per le quali esistono infinite superfici minimali distinte. Questo solleva la questione di quale superficie si formerà fisicamente: la risposta dipende da considerazioni di stabilità.

2.8 La Membrana Elastica: Equazione di Poisson

Consideriamo una membrana elastica tesa su un bordo fisso e soggetta a una forza distribuita. L'energia elastica della membrana deformata è proporzionale alla sua area superficiale, mentre il lavoro compiuto dalla forza esterna è proporzionale all'integrale della deformazione moltiplicato per la densità di forza.

Se $u(x, y)$ rappresenta lo spostamento verticale della membrana e $f(x, y)$ è la densità di forza (forza per unità di area), l'energia totale è:

$$E[u] = \sigma \iint \sqrt{1 + u_x^2 + u_y^2} \, dx \, dy - \iint f u \, dx \, dy$$

dove σ è la tensione superficiale della membrana. Per piccole deformazioni, possiamo approssimare $\sqrt{1 + u_x^2 + u_y^2} \approx 1 + \frac{1}{2}(u_x^2 + u_y^2)$, ottenendo:

$$E[u] \approx \sigma/2 \iint (u_x^2 + u_y^2) \, dx \, dy - \iint f u \, dx \, dy$$

Applicando il calcolo variazionale, l'equazione di Eulero-Lagrange è:

$$-\sigma(\partial^2 u / \partial x^2 + \partial^2 u / \partial y^2) = f$$

ovvero:

$$\nabla^2 u = -f/\sigma$$

Questa è l'equazione di Poisson, una delle equazioni più importanti della fisica matematica. Appare in innumerevoli contesti: elettrostatica (dove u è il potenziale elettrico e f è la densità di carica), gravitazione (u è il potenziale gravitazionale, f è la densità di massa), trasferimento di calore (u è la temperatura, f è la sorgente di calore), e molti altri.

Il caso speciale $f = 0$ dà l'equazione di Laplace:

$$\nabla^2 u = 0$$

Le soluzioni dell'equazione di Laplace sono chiamate funzioni armoniche e hanno proprietà notevoli: soddisfano il principio del valore medio (il valore in un punto è la media dei valori su qualsiasi sfera centrata in quel punto) e il principio del massimo (non possono avere massimi o minimi locali all'interno del dominio).

2.9 La Trave Vibrante: Equazione di Eulero-Bernoulli

Una trave sottile e flessibile, come una sbarra metallica o un ponte sospeso, vibra quando viene disturbata. La forma di queste vibrazioni è governata da un principio variazionale che bilancia energia cinetica ed energia potenziale elastica.

Consideriamo una trave di lunghezza L , densità lineare ρ (massa per unità di lunghezza), e rigidità flessionale EI (dove E è il modulo di Young e I è il momento d'inerzia della sezione trasversale). Se $u(x, t)$ rappresenta lo spostamento trasversale della trave nella posizione x al tempo t , l'energia cinetica è:

$$T = \frac{1}{2} \int_0^L \rho (\partial u / \partial t)^2 \, dx$$

e l'energia potenziale elastica (energia di deformazione) è:

$$V = \frac{1}{2} \int_0^L EI (\partial^2 u / \partial x^2)^2 dx$$

Il principio di Hamilton afferma che la traiettoria fisica del sistema è quella che rende stazionaria l'azione:

$$S = \int (T - V) dt$$

Applicando il calcolo variazionale a questo funzionale (ora un funzionale di una funzione di due variabili), otteniamo l'equazione di Eulero-Bernoulli:

$$\rho \partial^2 u / \partial t^2 + EI \partial^4 u / \partial x^4 = 0$$

Questa è un'equazione differenziale parziale del quarto ordine nello spazio e del secondo ordine nel tempo. Le soluzioni hanno la forma di onde che si propagano lungo la trave, con una relazione di dispersione particolare che lega frequenza e lunghezza d'onda.

Per trovare i modi normali di vibrazione, cerchiamo soluzioni della forma $u(x, t) = \varphi(x) \cos(\omega t)$, dove ω è la frequenza angolare. Sostituendo nell'equazione di Eulero-Bernoulli:

$$d^4 \varphi / dx^4 = (\rho \omega^2 / EI) \varphi$$

Questa è un'equazione agli autovalori. Per una trave con estremità semplicemente appoggiate ($\varphi = 0$ e $d^2 \varphi / dx^2 = 0$ alle estremità), le soluzioni sono:

$$\varphi_n(x) = \sin(n\pi x / L) \quad \omega_n = (n\pi / L)^2 \sqrt{EI / \rho}$$

dove $n = 1, 2, 3, \dots$. Questi sono i modi normali della trave vibrante. La frequenza fondamentale ($n = 1$) determina il tono principale che sentiamo quando la trave vibra, mentre i modi superiori ($n > 1$) producono gli armonici.

2.10 Il Problema del Controllo Ottimo

Il calcolo variazionale trova applicazione moderna nella teoria del controllo ottimo. Supponiamo di avere un sistema dinamico descritto dall'equazione differenziale:

$$dx/dt = f(x, u, t)$$

dove $x(t)$ è lo stato del sistema e $u(t)$ è il controllo (l'input che possiamo manipolare). Vogliamo trovare il controllo $u(t)$ che minimizza un certo funzionale di costo, tipicamente della forma:

$$J[u] = \int_{t_0}^{t_1} L(x, u, t) dt + \Phi(x(t_1))$$

dove L è il costo istantaneo e Φ è il costo terminale.

Questo è un problema variazionale con vincolo (il vincolo è l'equazione dinamica). La soluzione può essere trovata usando il principio del massimo di Pontryagin, che è una generalizzazione delle equazioni di Eulero-Lagrange per problemi con vincoli dinamici.

Introduciamo le variabili aggiuntive (variabili aggiunte o co-stati) $\lambda(t)$ e definiamo l'hamiltoniana:

$$H(x, u, \lambda, t) = L(x, u, t) + \lambda^T f(x, u, t)$$

Il principio del massimo afferma che il controllo ottimo $u^*(t)$ minimizza H rispetto a u , e che le variabili di stato e aggiuntive soddisfano le equazioni:

$$dx/dt = \partial H / \partial \lambda \quad d\lambda/dt = -\partial H / \partial x$$

con condizioni al bordo $x(t_0) = x_0$ (condizione iniziale) e $\lambda(t_1) = \partial \Phi / \partial x|_{x(t_1)}$ (condizione terminale).

Un esempio classico è il problema del tempo minimo: vogliamo portare un sistema da uno stato iniziale a uno stato finale nel minor tempo possibile. Se il controllo è limitato ($|u| \leq u_{\max}$), la soluzione ottimale è spesso un controllo "bang-bang": $u = \pm u_{\max}$, passando bruscamente da un estremo all'altro in certi istanti di commutazione.

2.11 La Geodesia su Superfici di Curvatura Costante

Le superfici di curvatura gaussiana costante hanno proprietà geometriche speciali, e le loro geodetiche possono essere studiate in modo sistematico. Esistono tre tipi fondamentali:

1. Curvatura $K = 0$ (Superfici Piatte) Il piano è l'esempio più ovvio. Ma esistono altre superfici piatte, come il cilindro e il cono. Le geodetiche su un cilindro possono essere linee rette lungo l'asse, cerchi perpendicolari all'asse, o eliche (spirali che avvolgono il cilindro con angolo costante).

2. Curvatura $K > 0$ (Superfici Ellittiche) La sfera è il caso principale. Le geodetiche sono cerchi massimi, e due geodetiche qualsiasi si intersecano sempre (a differenza del piano, dove linee parallele non si incontrano mai). Questo è alla base della geometria ellittica o di Riemann.

3. Curvatura $K < 0$ (Superfici Iperboliche) La pseudosfera (generata dalla rotazione della trattrice) e il paraboloide iperbolico ("superficie a sella") sono esempi. Le geodetiche hanno comportamenti più complessi: date una geodetica e un punto non su di essa, esistono infinite geodetiche passanti per quel punto che non intersecano la geodetica originale. Questo è alla base della geometria iperbolica o di Lobačevskij.

Carl Friedrich Gauss scoprì il teorema egregium ("notevole"), che afferma che la curvatura gaussiana di una superficie è una proprietà intrinseca: può essere determinata solo da misurazioni sulla superficie stessa, senza riferimento allo spazio ambiente. Questo significa che se deformiamo una superficie senza allungarla o comprimerla (una deformazione isometrica), la curvatura gaussiana rimane invariata.

2.12 L'Energia di una Curva: Elastica

Una sbarra elastica sottile (come un righello di metallo flessibile) che viene piegata immagazzina energia elastica proporzionale al quadrato della sua curvatura. Se $\kappa(s)$ è la curvatura in funzione della lunghezza d'arco s , l'energia elastica totale è:

$$E = \frac{1}{2} \int \kappa^2 ds$$

Le forme di equilibrio di una sbarra elastica soggetta a vincoli agli estremi sono le curve che minimizzano questa energia. Queste curve, chiamate elastiche, furono studiate da Jacob Bernoulli, Leonhard Euler e altri.

La curvatura κ di una curva piana $y(x)$ è:

$$\kappa = y'' / (1 + y'^2)^{3/2}$$

Per piccole deformazioni (y' piccolo), possiamo approssimare $\kappa \approx y''$, e l'energia diventa:

$$E \approx \frac{1}{2} \int (y'')^2 dx$$

L'equazione di Eulero-Lagrange per questo funzionale è:

$$d^4y/dx^4 = 0$$

che ha come soluzioni polinomi cubici. Questi rappresentano la forma approssimata di una trave elastica piegata.

Per deformazioni grandi, l'analisi completa delle elastiche è più sottile e porta a integrali ellittici. Le elastiche possono assumere forme affascinanti, incluse curve chiuse come lemniscate (a forma di otto).

2.13 Campi Scalari e l'Equazione di Eulero-Lagrange in Più Dimensioni

Molti problemi fisici coinvolgono campi definiti su regioni dello spazio bidimensionale o tridimensionale. Per esempio, la distribuzione di temperatura in una piastra metallica, il potenziale elettrico attorno a una configurazione di cariche, o il campo gravitazionale in una regione di spazio.

Consideriamo un funzionale che dipende da un campo scalare $\phi(x, y)$ e dai suoi gradienti:

$$I[\phi] = \iint F(x, y, \phi, \partial\phi/\partial x, \partial\phi/\partial y) dx dy$$

L'equazione di Eulero-Lagrange in questo caso è:

$$\partial F / \partial \phi - \partial / \partial x (\partial F / \partial \phi_x) - \partial / \partial y (\partial F / \partial \phi_y) = 0$$

dove $\phi_x = \partial\phi/\partial x$ e $\phi_y = \partial\phi/\partial y$.

Un esempio importante è il funzionale di Dirichlet:

$$I[\phi] = \frac{1}{2} \iint [(\partial\phi/\partial x)^2 + (\partial\phi/\partial y)^2] dx dy$$

che rappresenta l'energia di un campo. Qui $F = \frac{1}{2}(\phi_x^2 + \phi_y^2)$, quindi:

$$\partial F / \partial \varphi = 0 \quad \partial F / \partial \varphi_x = \varphi_x \quad \partial F / \partial \varphi_y = \varphi_y$$

L'equazione di Eulero-Lagrange diventa:

$$-\partial^2 \varphi / \partial x^2 - \partial^2 \varphi / \partial y^2 = 0$$

che è esattamente l'equazione di Laplace $\nabla^2 \varphi = 0$. Questo mostra che le funzioni armoniche sono quelle che minimizzano il funzionale di Dirichlet, ovvero l'energia del campo.

In tre dimensioni, per un campo $\varphi(x, y, z)$, il funzionale di Dirichlet diventa:

$$I[\varphi] = \frac{1}{2} \iiint |\nabla \varphi|^2 dx dy dz = \frac{1}{2} \iiint [(\partial \varphi / \partial x)^2 + (\partial \varphi / \partial y)^2 + (\partial \varphi / \partial z)^2] dx dy dz$$

e l'equazione di Eulero-Lagrange è ancora l'equazione di Laplace tridimensionale.

2.14 Il Principio di Minima Azione in Meccanica

Il principio di minima azione, o principio di Hamilton, è forse l'applicazione più profonda e generale del calcolo variazionale alla fisica. Afferma che la traiettoria fisica di un sistema meccanico è quella che rende stazionaria (solitamente minimizza) l'azione.

Per un sistema con coordinate generalizzate $q_i(t)$, l'azione è definita come:

$$S = \int_{(t_1 \text{ a } t_2)} L(q, \dot{q}, t) dt$$

dove $L = T - V$ è la lagrangiana (differenza tra energia cinetica T ed energia potenziale V), e $\dot{q}$ indica la derivata temporale di q .

Applicando il calcolo variazionale, otteniamo le equazioni di Eulero-Lagrange della meccanica:

$$d/dt(\partial L / \partial \dot{q}_i) - \partial L / \partial q_i = 0$$

per ogni coordinata i . Queste equazioni sono equivalenti alle equazioni di Newton, ma hanno il vantaggio di essere applicabili in qualsiasi sistema di coordinate e di trattare automaticamente i vincoli.

Consideriamo un esempio semplice: una particella di massa m in un potenziale $V(x)$. La lagrangiana è:

$$L = \frac{1}{2} m \dot{x}^2 - V(x)$$

L'equazione di Eulero-Lagrange è:

$$d/dt(m \dot{x}) - (-dV/dx) = 0$$

ovvero:

$$m \ddot{x} = -dV/dx$$

che è esattamente la seconda legge di Newton $F = ma$, con $F = -dV/dx$ (la forza è il negativo del gradiente del potenziale).

Il principio di minima azione non è solo un modo elegante di riformulare la meccanica newtoniana. Ha profonde implicazioni filosofiche: suggerisce che la natura "sceglie" le traiettorie secondo un principio globale di ottimizzazione, piuttosto che seguire leggi locali istante per istante. Inoltre, questa formulazione si generalizza naturalmente alla meccanica quantistica (attraverso l'integrale di Feynman sui cammini) e alla teoria dei campi, dove diventa il punto di partenza per formulare tutte le teorie fisiche fondamentali.

2.15 Teorema di Noether: Simmetria e Conservazione

Emmy Noether, nel 1918, scoprì una connessione profonda tra simmetrie e leggi di conservazione che è centrale nella fisica moderna. Il suo teorema afferma che per ogni simmetria continua di un sistema fisico (una trasformazione che lascia invariata l'azione), esiste una corrispondente quantità conservata.

Consideriamo una lagrangiana $L(q, \dot{q}, t)$. Supponiamo che esista una trasformazione infinitesima:

$$q_i \rightarrow q_i + \varepsilon f_i(q, t) \quad t \rightarrow t + \varepsilon g(q, t)$$

dove ε è un parametro infinitesimo, tale che la lagrangiana cambi al più per una derivata totale (il che lascia l'azione invariata). Allora la quantità:

$$Q = \sum_i (\partial L / \partial \dot{q}_i) f_i - L g$$

è conservata lungo le traiettorie fisiche ($dQ/dt = 0$).

Vediamo alcuni esempi fondamentali:

Invarianza per traslazioni spaziali → Conservazione del momento Se la lagrangiana non dipende esplicitamente dalla posizione (omogeneità dello spazio), allora il momento $p = \partial L / \partial \dot{q}$ è conservato.

Invarianza per traslazioni temporali → Conservazione dell'energia Se la lagrangiana non dipende esplicitamente dal tempo, allora l'energia totale $E = \sum_i \dot{q}_i (\partial L / \partial \dot{q}_i) - L$ (l'hamiltoniana) è conservata.

Invarianza per rotazioni → Conservazione del momento angolare Se la lagrangiana è invariante sotto rotazioni (isotropia dello spazio), allora il momento angolare $L = \mathbf{r} \times \mathbf{p}$ è conservato.

Il teorema di Noether unifica tutte le leggi di conservazione sotto un unico principio: sono conseguenze di simmetrie. Questo risultato è di importanza fondamentale non solo nella meccanica classica, ma anche nella teoria quantistica dei campi, dove le simmetrie di gauge portano alle interazioni fondamentali della natura.

In teoria dei campi, il teorema di Noether associa a ogni simmetria continua una corrente conservata (un campo vettoriale la cui divergenza è zero). Per esempio, la simmetria di

gauge locale dell'elettrodinamica porta alla conservazione della carica elettrica, espressa dall'equazione di continuità $\partial \rho / \partial t + \nabla \cdot \mathbf{J} = 0$, dove ρ è la densità di carica e $\mathbf{J}$ è la densità di corrente.

Conclusione della Parte I

Il calcolo variazionale fornisce un framework matematico potente e unificante per comprendere una vasta gamma di fenomeni naturali. Dai percorsi di luce alle forme di equilibrio delle strutture elastiche, dalle vibrazioni delle travi alle leggi fondamentali della meccanica, i principi variazionali emergono ripetutamente come descrizione naturale dei sistemi fisici.

La bellezza del calcolo variazionale risiede nella sua capacità di formulare problemi fisici in termini di ottimizzazione: la natura sembra "scegliere" configurazioni che minimizzano (o rendono stazionarie) certe quantità funzionali. Questo approccio non solo fornisce un metodo sistematico per risolvere problemi complessi, ma rivela anche connessioni profonde tra fenomeni apparentemente diversi.

I quindici problemi che abbiamo esaminato in dettaglio rappresentano solo un assaggio della ricchezza del calcolo variazionale. Dalle geodetiche sulle superfici curve alle elastiche, dall'equazione di Poisson al teorema di Noether, abbiamo visto come un singolo framework matematico possa descrivere la geometria, la meccanica, l'elasticità e molto altro.

Nella seconda parte di questo trattato, vedremo come questi principi astratti si manifestano concretamente nei sistemi biologici e naturali. Le api che costruiscono alveari esagonali, le piante che dispongono le loro foglie secondo spirali auree, i ragni che tessono ragnatele ottimali: tutti questi fenomeni possono essere compresi attraverso il prisma dell'ottimizzazione che abbiamo sviluppato in questa prima parte. La matematica del calcolo variazionale non è solo uno strumento teorico, ma la lingua stessa con cui la natura esprime le sue soluzioni ai problemi che deve affrontare.

PARTE II: OTTIMIZZAZIONE IN NATURA

Capitolo 3: La Geometria degli Alveari

3.1 Il Problema dell'Ape: Massimizzare il Volume, Minimizzare il Materiale

L'alveare delle api rappresenta uno degli esempi più straordinari di ottimizzazione geometrica in natura. Quando le api costruiscono le celle per immagazzinare miele e allevare larve, devono risolvere un problema di ingegneria complesso: creare strutture che massimizzino lo spazio interno utilizzando la minima quantità di cera. La cera è una risorsa preziosa per le api, poiché la sua produzione richiede un grande dispendio energetico. Si stima che le api debbano consumare circa otto chilogrammi di miele per produrre un solo chilogrammo di cera.

La soluzione trovata dalle api è la struttura esagonale. Le celle dell'alveare sono prismi esagonali regolari disposti in modo da condividere le pareti, creando una struttura a nido d'ape. Questa configurazione non è casuale, ma rappresenta la soluzione ottimale a un preciso problema matematico.

Consideriamo il problema in due dimensioni prima di affrontare la struttura tridimensionale completa. Supponiamo di voler dividere un piano in celle di uguale area, utilizzando la minima lunghezza totale di perimetro. Questo è un problema isoperimetrico generalizzato: date n celle, ciascuna di area A , quale disposizione minimizza il perimetro totale?

La risposta matematica a questo problema fu congetturata già nell'antichità (è nota come la "congettura del favo") ma fu dimostrata rigorosamente solo nel 1999 dal matematico Thomas Hales. Il teorema afferma che la configurazione ottimale è esattamente quella esagonale utilizzata dalle api.

Per comprendere perché gli esagoni sono ottimali, confrontiamo diverse geometrie regolari. Se vogliamo piastrellare il piano con poligoni regolari tutti uguali, solo tre forme funzionano: triangoli equilateri, quadrati ed esagoni regolari. Calcoliamo il rapporto tra perimetro e area per ciascuna forma.

Per un triangolo equilatero di lato l :

- Area: $A = (\sqrt{3}/4)l^2$
- Perimetro: $P = 3l$
- Rapporto $P/\sqrt{A} = 3l/\sqrt{[(\sqrt{3}/4)l^2]} = 3l/(l\sqrt{(\sqrt{3}/4)}) = 6/\sqrt{(\sqrt{3})} \approx 4,559$

Per un quadrato di lato l :

- Area: $A = l^2$
- Perimetro: $P = 4l$
- Rapporto $P/\sqrt{A} = 4l/l = 4$

Per un esagono regolare di lato l :

- Area: $A = (3\sqrt{3}/2)l^2$
- Perimetro: $P = 6l$
- Rapporto $P/\sqrt{A} = 6l/\sqrt{[(3\sqrt{3}/2)l^2]} = 6l/(l\sqrt{(3\sqrt{3}/2)}) \approx 3,722$

Vediamo che l'esagono ha il rapporto più basso, quindi utilizza la minor quantità di perimetro per una data area. Questo risultato si può comprendere anche considerando che l'esagono regolare è il poligono regolare che più si avvicina al cerchio (che ha il rapporto perimetro-area ottimale assoluto) tra quelli che possono piastrellare il piano.

3.2 La Struttura Tridimensionale delle Celle

La realtà delle celle dell'alveare è ancora più sofisticata della semplice struttura bidimensionale. Le celle non sono cilindri esagonali semplici, ma hanno una base particolare che aggiunge un ulteriore livello di ottimizzazione.

Ogni cella dell'alveare è chiusa sul fondo da una struttura composta da tre rombi. Questi rombi sono disposti in modo tale che il vertice dove si incontrano i tre rombi sporge verso l'interno della cella. Questa configurazione serve due scopi: prima di tutto, fornisce rinforzo strutturale; in secondo luogo, la geometria è tale che quando le celle sono costruite su entrambi i lati del foglio di cera (creando due strati di celle back-to-back), le sporgenze di un lato si incastrano perfettamente nelle depressioni dell'altro lato, massimizzando l'utilizzo dello spazio.

Il matematico e fisico René-Antoine Ferchault de Réaumur studiò questa struttura nel diciottesimo secolo e pose a degli matematici il seguente problema: qual è l'angolo ottimale dei rombi che minimizza la superficie totale della cella (e quindi la quantità di cera necessaria) per un dato volume?

Il matematico Niklaus I Bernoulli risolse il problema trovando che gli angoli ottimali dei rombi dovrebbero essere di circa 70,53 gradi e 109,47 gradi. Quando gli naturalisti misurarono gli angoli reali nelle celle costruite dalle api, trovarono valori straordinariamente vicini: circa 70 gradi e 110 gradi. La piccola discrepanza era probabilmente dovuta alle imprecisioni nelle misurazioni dell'epoca.

Formuliamo matematicamente il problema. Una cella esagonale ha sei pareti laterali rettangolari e una base composta da tre rombi. Se h è l'altezza delle pareti laterali, l è il lato dell'esagono, e θ è uno degli angoli dei rombi (l'altro è necessariamente $180^\circ - \theta$), la superficie totale della cella può essere espressa come funzione di questi parametri.

La superficie delle sei pareti laterali è: $S_{\text{laterale}} = 6lh$

Per calcolare la superficie della base romboidale, dobbiamo considerare la geometria tridimensionale. Ogni rombo ha un lato di lunghezza l e angoli θ e $180^\circ - \theta$. La superficie di un rombo con lato l e angolo θ è: $S_{\text{rombo}} = l^2 \sin(\theta)$

La base consiste di tre tali rombi, quindi: $S_{\text{base}} = 3l^2 \sin(\theta)$

Ma c'è un vincolo: il volume della cella è fissato. Il volume è determinato dall'area esagonale di base moltiplicata per un'altezza effettiva che dipende da θ . La geometria è sottile, ma il vincolo di volume fisso lega h e θ .

Senza entrare in tutti i dettagli geometrici (che richiedono considerazioni trigonometriche tridimensionali), il risultato finale è che minimizzando la superficie totale rispetto a θ , si ottiene:

$$\theta_{\text{ottimale}} = \arccos(1/3) \approx 70,53^\circ$$

Questo valore è notevolmente vicino a quello osservato nelle celle reali delle api. È straordinario che un insetto, attraverso processi evolutivi e istintivi, sia arrivato a una soluzione che corrisponde così precisamente al risultato di un'analisi matematica sofisticata.

3.3 L'Ottimizzazione Termica dell'Alveare

Oltre all'ottimizzazione geometrica delle singole celle, la struttura complessiva dell'alveare presenta caratteristiche che ottimizzano la termoregolazione. Le api devono mantenere la temperatura all'interno dell'alveare in un range stretto, tipicamente tra 34 e 36 gradi Celsius, per garantire il corretto sviluppo delle larve.

La forma complessiva dell'alveare tende a essere sferica o ellissoidale. Questa geometria non è casuale: per un dato volume, la sfera ha la superficie minima. Minimizzare la superficie significa minimizzare la perdita di calore verso l'esterno, riducendo così l'energia che le api devono spendere per mantenere la temperatura interna.

Il rapporto superficie-volume per una sfera di raggio r è: $S/V = 4\pi r^2 / (4\pi r^3/3) = 3/r$

Per confronto, un cubo di lato a ha: $S/V = 6a^2/a^3 = 6/a$

A parità di volume ($V = a^3 = 4\pi r^3/3$), il cubo ha un rapporto superficie-volume maggiore del 24% rispetto alla sfera. Questo significa che un alveare cubico perderebbe calore significativamente più velocemente di uno sferico.

Inoltre, le api utilizzano strategicamente lo spazio all'interno dell'alveare. Le celle contenenti covata (uova e larve) sono posizionate al centro, dove la temperatura è più stabile e più facile da controllare. Le celle contenenti miele e polline sono disposte in anelli concentrici attorno alla zona di covata, fungendo da isolamento termico aggiuntivo.

3.4 Il Problema del Matematico: Dimostrazione di Hales

La dimostrazione completa della congettura del favo da parte di Thomas Hales nel 1999 rappresenta un trionfo della matematica moderna. La dimostrazione è estremamente tecnica e si basa su metodi di ottimizzazione combinatoria e geometria computazionale.

L'approccio di Hales consiste nel dimostrare che qualsiasi configurazione di celle diversa da quella esagonale può essere migliorata (ossia, si può ridurre il perimetro mantenendo la stessa area totale) attraverso una serie di trasformazioni locali. Il passo chiave è dimostrare che ogni configurazione non esagonale contiene necessariamente una "zona di inefficienza" che può essere eliminata.

La dimostrazione utilizza tecniche di programmazione lineare per verificare che in tutte le possibili configurazioni locali di celle, quella esagonale ha il perimetro minimo. Data la complessità combinatoria del problema, parte della verifica è stata eseguita con l'aiuto del computer, rendendo questa una delle prime dimostrazioni assistite da computer di un teorema geometrico importante.

Il risultato di Hales conferma matematicamente ciò che le api "sapevano" già da milioni di anni: la struttura esagonale è veramente ottimale.

Capitolo 4: Filotassi e la Spirale Aurea

4.1 La Disposizione delle Foglie: Massimizzare l'Esposizione al Sole

Quando osserviamo attentamente le piante, notiamo che le foglie non sono disposte casualmente lungo il fusto. Esiste invece un ordine geometrico preciso, chiamato filotassi (dal greco "phyllon" = foglia e "taxis" = disposizione). Questa disposizione non è arbitraria, ma riflette una strategia di ottimizzazione: massimizzare l'esposizione alla luce solare di ogni foglia, minimizzando l'ombreggiamento reciproco.

Il modello più comune di filotassi è quello a spirale. Immaginiamo di guardare il fusto dall'alto verso il basso e di numerare le foglie in sequenza. L'angolo tra una foglia e la successiva (misurato attorno al fusto) tende a essere costante e assume valori specifici legati alla sequenza di Fibonacci.

La sequenza di Fibonacci è definita ricorsivamente: $F_0 = 0$, $F_1 = 1$, $F_n = F_{(n-1)} + F_{(n-2)}$

Quindi: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

Il rapporto tra numeri consecutivi di Fibonacci converge al numero aureo: $\phi = (1 + \sqrt{5})/2 \approx 1,618...$

che soddisfa l'equazione $\phi^2 = \phi + 1$, o equivalentemente $\phi = 1 + 1/\phi$.

L'angolo di divergenza più comune in filotassi è l'angolo aureo: $\alpha = 360^\circ/\phi^2 = 360^\circ(2 - \phi) \approx 137,5^\circ$

Questo angolo ha una proprietà notevole: è il numero irrazionale "più irrazionale" nel senso che la sua frazione continua ha tutti i termini uguali a 1: $\phi = 1 + 1/(1 + 1/(1 + 1/(1 + ...)))$

4.2 Perché 137,5 Gradi è Ottimale

Per comprendere perché l'angolo aureo ottimizza la disposizione delle foglie, consideriamo il problema astratto di disporre punti su un cilindro (che rappresenta il fusto della pianta). Vogliamo che i punti siano distribuiti nel modo più uniforme possibile, evitando allineamenti e raggruppamenti.

Supponiamo che le foglie successive siano separate da un angolo costante α . Dopo n foglie, avremo fatto un certo numero di giri completi attorno al fusto più una frazione rimanente. Se $\alpha/360^\circ = p/q$ (dove p e q sono interi), allora dopo q foglie torneremo esattamente alla posizione angolare iniziale, creando q linee verticali di foglie lungo il fusto.

Queste linee verticali sono chiamate parastiche, e la loro presenza indica periodicità nella disposizione. La periodicità è indesiderabile perché crea "corsie" di spazio vuoto tra le parastiche, riducendo l'efficienza dell'impaccamento.

L'angolo aureo è speciale perché il rapporto $\alpha/360^\circ$ è il numero irrazionale con la peggiore approssimazione razionale. Matematicamente, questo significa che l'angolo aureo minimizza la periodicità, creando la distribuzione più uniforme possibile.

Possiamo formulare questo come un problema di ottimizzazione. Definiamo la discrepanza D di una sequenza di n punti su un cerchio come la differenza massima tra la frazione di punti che cadono in un arco qualsiasi e la frazione che ci aspetteremmo se i punti fossero distribuiti uniformemente.

Per una sequenza generata da un angolo α , la discrepanza cresce al tasso minimo possibile quando α è l'angolo aureo. Questo risultato, dimostrato nella teoria dei numeri, spiega matematicamente perché le piante hanno evoluto questo particolare angolo.

4.3 Le Spirali nei Girasoli e nelle Pigne

Uno dei fenomeni più affascinanti in botanica è l'apparizione di spirali multiple nei girasoli, nelle pigne, negli ananas e in molte altre strutture vegetali. Se contiamo il numero di spirali in senso orario e antiorario, troviamo invariabilmente numeri consecutivi della sequenza di Fibonacci.

Un tipico girasole mostra 21 spirali in una direzione e 34 nell'altra (o 34 e 55 per girasoli più grandi, o addirittura 55 e 89 per esemplari molto grandi). Le pigne mostrano comunemente 8 e 13 spirali, o 5 e 8.

Questo fenomeno emerge naturalmente da un modello matematico semplice proposto dai botanici. Immaginiamo che i semi di un girasole siano generati sequenzialmente al centro e poi spinti verso l'esterno man mano che nuovi semi vengono generati. Ogni seme emerge con un angolo pari all'angolo aureo rispetto al precedente.

Possiamo modellare questo processo in coordinate polari. Il seme numero n ha coordinate:
 $r_n = c\sqrt{n}$ $\theta_n = n \times \alpha$

dove c è una costante che determina quanto rapidamente i semi si allontanano dal centro, e $\alpha \approx 137,5^\circ$ è l'angolo aureo.

Il fattore $\sqrt{n}$ per il raggio deriva da considerazioni di area: se i semi occupano area approssimativamente costante, e sono distribuiti uniformemente in termini angolari, allora l'area spazzata deve crescere linearmente con n , il che implica che r^2 deve crescere linearmente con n , quindi r proporzionale a $\sqrt{n}$.

Questo modello, noto come modello di Vogel (dal botanico H. Vogel che lo propose nel 1979), genera pattern che corrispondono straordinariamente bene a quelli osservati in natura. Le spirali che vediamo emergono automaticamente dalla geometria: connettendo ogni seme con il suo "vicino più prossimo" in una certa direzione, si tracciano naturalmente spirali, e il numero di queste spirali è necessariamente un numero di Fibonacci.

4.4 La Matematica delle Spirali di Fibonacci

Per comprendere perché emergono specificamente i numeri di Fibonacci, dobbiamo analizzare la struttura matematica più in profondità. Consideriamo i semi disposti secondo il modello di Vogel. Due semi sono "vicini" se la loro separazione angolare, ridotta modulo 360° , è piccola.

Dopo F_n semi (dove F_n è un numero di Fibonacci), l'angolo totale percorso è: $\theta = F_n \times 137,5^\circ$

Utilizzando le proprietà del numero aureo, si può dimostrare che questo angolo, ridotto modulo 360° , è molto vicino a un multiplo di 360° . Più precisamente, la frazione: $F_n \times \varphi / F_{(n+1)}$

converge a 1 estremamente rapidamente. Questo significa che dopo F_n semi, siamo quasi tornati alla posizione angolare iniziale, creando una parastiche "quasi chiusa".

Le vere parastiche visibili sono quelle corrispondenti ai numeri di Fibonacci perché queste sono le approssimazioni razionali dell'angolo aureo fornite dai convergenti della sua frazione continua: $F_n / F_{(n+1)} \rightarrow 1/\varphi$

Ogni convergente fornisce la "migliore" approssimazione razionale con denominatore fino a $F_{(n+1)}$, e queste approssimazioni corrispondono alle parastiche più prominenti.

4.5 Ottimizzazione Evolutiva

La prevalenza della filotassi aurea nelle piante non è coincidenza, ma risultato di pressione evolutiva. Le piante che dispongono le loro foglie secondo l'angolo aureo hanno un vantaggio selettivo: massimizzano l'intercettazione della luce solare.

Modelli computazionali hanno confermato questa ipotesi. Quando si simula la crescita di piante con diversi angoli di divergenza e si calcola la quantità totale di luce catturata (tenendo conto dell'ombreggiamento reciproco delle foglie), l'angolo aureo emerge come ottimale o quasi-ottimale in un'ampia gamma di condizioni.

È importante notare che l'ottimalità è robusta: angoli vicini all'angolo aureo funzionano quasi altrettanto bene, il che spiega perché in natura si osservano piccole variazioni attorno al valore teorico di $137,5^\circ$.

Inoltre, la filotassi aurea non è l'unico pattern osservato in natura. Alcune piante utilizzano disposizioni diverse (come la filotassi opposta o verticillata) che sono ottimali per le loro specifiche esigenze ecologiche. Tuttavia, per piante con crescita a spirale continua, l'angolo aureo rappresenta la soluzione evolutivamente stabile.

Capitolo 5: Ragnatele e Ottimizzazione Strutturale

5.1 Il Problema del Ragno: Massima Cattura con Minima Seta

I ragni tessitori di orb-webs (ragnatele circolari) sono maestri ingegneri che risolvono quotidianamente un complesso problema di ottimizzazione: costruire una trappola efficace per catturare insetti utilizzando la minima quantità di seta. La seta è una risorsa costosa dal punto di vista metabolico, quindi i ragni devono essere estremamente efficienti nella sua utilizzazione.

Una tipica ragnatela circolare consiste di diverse componenti:

1. **Fili radiali:** partono dal centro e si estendono verso l'esterno come raggi di una ruota

2. **Spirale di cattura:** un filo appiccicoso che forma una spirale logaritmica connettendo i fili radiali
3. **Fili di ancoraggio:** connettono la ragnatela a supporti esterni (rami, foglie, ecc.)
4. **Hub centrale:** la zona centrale dove il ragno attende

La geometria di questa struttura non è casuale. I fili radiali forniscono supporto strutturale e tensione, mentre la spirale appiccicosa cattura le prede. La disposizione radiale ottimizza la distribuzione delle forze: quando un insetto colpisce la ragnatela, l'impatto viene distribuito uniformemente su tutti i fili radiali, minimizzando la probabilità di rottura.

5.2 La Geometria Ottimale: Spirale Logaritmica vs Spirale Archimedeana

La spirale di cattura in molte ragnatele non è una spirale di Archimede (dove la distanza dal centro cresce linearmente con l'angolo), ma una spirale logaritmica (dove la distanza dal centro cresce esponenzialmente con l'angolo).

Una spirale logaritmica in coordinate polari ha l'equazione: $r = a e^{(b\theta)}$

dove a e b sono costanti. Questa curva ha una proprietà notevole: l'angolo tra la curva e il raggio vettore in ogni punto è costante. In altre parole, la spirale "taglia" tutti i raggi con lo stesso angolo.

Per una ragnatela, questo significa che la spirale di cattura interseca ogni filo radiale con lo stesso angolo, creando una struttura geometricamente uniforme. Questa uniformità è importante per due ragioni:

1. **Distribuzione ottimale della tensione:** L'angolo costante assicura che la tensione nei fili appiccicosi sia distribuita uniformemente, massimizzando la resistenza complessiva.
2. **Deteção efficiente:** Il ragno, posizionato al centro, può rilevare vibrazioni da qualsiasi direzione con uguale efficacia, poiché il percorso del segnale lungo i fili radiali è geometricamente equivalente per ogni direzione.

In realtà, molte ragnatele mostrano una spirale che è approssimativamente logaritmica nella zona esterna, ma può deviare da questo pattern nella zona più interna. Questo riflette un compromesso tra diverse esigenze: nella zona centrale, dove gli insetti catturati sono più piccoli e le forze sono minori, il ragno può permettersi di deviare dall'ottimale geometrico per risparmiare seta.

5.3 Spaziatura Ottimale dei Fili

La distanza tra le spirali successive (la "maglia" della ragnatela) è un parametro critico. Una maglia troppo larga lascia passare piccoli insetti; una troppo stretta richiede troppa seta senza benefici proporzionali nella cattura.

Studi empirici hanno mostrato che la spaziatura è ottimizzata in funzione della dimensione tipica delle prede. Per ragni che cacciano principalmente piccoli insetti come moscerini, la

maglia è più fitta (2-3 mm). Per ragni che cacciano prede più grandi, la maglia può essere di 5-10 mm.

Possiamo modellare questo come un problema di ottimizzazione stocastica. Supponiamo che gli insetti di dimensione d arrivino con una certa distribuzione di probabilità $P(d)$. La probabilità che un insetto di dimensione d venga catturato da una ragnatela con maglia s è:

$$p_{\text{cattura}}(d, s) = 1 - e^{(-cd/s)}$$

dove c è una costante che dipende dalla geometria della ragnatela e dalle proprietà adesive della seta. Questa forma esponenziale cattura il fatto che insetti molto più piccoli della maglia hanno bassa probabilità di essere catturati, mentre insetti comparabili o più grandi della maglia sono quasi certamente catturati.

Il costo in termini di seta è proporzionale a $1/s$ (maglia più fitta richiede più seta). Il beneficio è proporzionale al tasso di cattura atteso:

$$B = \int P(d) p_{\text{cattura}}(d, s) V(d) dd$$

dove $V(d)$ è il valore nutrizionale di un insetto di dimensione d .

La spaziatura ottimale s^* massimizza il beneficio netto $B - k/s$, dove k è una costante che rappresenta il costo metabolico della seta. Derivando rispetto a s e ponendo uguale a zero, otteniamo una condizione per s^* che dipende dalla distribuzione delle prede e dal costo della seta.

5.4 Proprietà Meccaniche della Seta

La seta di ragno è un materiale straordinario con proprietà meccaniche eccezionali. Diversi tipi di seta hanno proprietà diverse, ottimizzate per funzioni specifiche:

Seta draglina (usata per i fili radiali e di sicurezza):

- Resistenza alla trazione: 1,1 GPa (comparabile all'acciaio)
- Elasticità: può allungarsi del 30-40% prima di rompersi
- Tenacità (energia assorbita prima della rottura): superiore al Kevlar

Seta vischiosa (usata per la spirale di cattura):

- Meno resistente della draglina
- Altamente elastica: può allungarsi fino al 270%
- Ricoperta di goccioline adesive

La combinazione di questi materiali crea una struttura ottimizzata per assorbire l'impatto di insetti volanti. Quando un insetto colpisce la ragnatela, l'energia cinetica viene dissipata in tre modi:

1. **Deformazione elastica:** I fili si allungano, immagazzinando energia elastica
2. **Dissipazione viscosa:** Le goccioline adesive si deformano, dissipando energia per attrito interno

3. **Distribuzione:** L'impatto viene distribuito su una vasta area della ragnatela attraverso i fili radiali

Modelli meccanici della ragnatela trattano la struttura come un sistema massa-molla non lineare. L'equazione del moto per un punto della ragnatela soggetto a un impatto può essere scritta come:

$$m \frac{d^2u}{dt^2} + \gamma \frac{du}{dt} + ku + \alpha u^3 = F(t)$$

dove u è lo spostamento, m è la massa effettiva, γ è il coefficiente di smorzamento viscoso, k è la costante elastica lineare, αu^3 rappresenta la non-linearità elastica (la seta diventa più rigida quando molto allungata), e $F(t)$ è la forza d'impatto.

Il termine non lineare αu^3 è cruciale: impedisce che la ragnatela si rompa sotto impatti forti, poiché la rigidità aumenta con la deformazione, limitando l'allungamento massimo.

5.5 Ottimizzazione Dinamica: Adattamento alla Disponibilità di Risorse

I ragni non costruiscono sempre ragnatele identiche. Adattano la dimensione, la densità della maglia e altre caratteristiche in risposta alle condizioni ambientali e alla loro condizione fisiologica.

Un ragno ben nutrito, con ampie riserve di seta, può permettersi di costruire una ragnatela più grande e con maglia più fitta. Un ragno affamato, con riserve limitate, costruirà una ragnatela più piccola e meno densa, ma posizionata strategicamente in luoghi dove la probabilità di cattura è massima.

Questo comportamento riflette una forma di ottimizzazione dinamica. Il ragno deve risolvere un problema di decisione sequenziale: data la sua attuale riserva di seta R e la distribuzione attesa di prede, quale configurazione di ragnatela massimizza il valore atteso di catture future?

Questo può essere formulato come un problema di programmazione dinamica. Definiamo $V(R)$ come il valore atteso delle catture future quando il ragno ha riserva R . Il ragno sceglie una configurazione di ragnatela caratterizzata da parametri (dimensione A , densità di maglia ρ) che massimizza:

$$E[\text{catture} \mid A, \rho] + \beta V(R - \text{costo}(A, \rho))$$

dove β è un fattore di sconto che rappresenta il valore delle risorse future rispetto a quelle presenti, e $\text{costo}(A, \rho)$ è la quantità di seta necessaria per quella configurazione.

Studi comportamentali hanno confermato che i ragni effettivamente modificano le loro ragnatele in modi consistenti con questa analisi teorica, anche se ovviamente non eseguono calcoli espliciti, ma seguono regole euristiche evolute attraverso selezione naturale.

Capitolo 6: Reti Vascolari e Ottimizzazione del Trasporto

6.1 La Legge di Murray: Diametro Ottimale dei Vasi Sanguigni

Il sistema circolatorio degli animali e il sistema vascolare delle piante affrontano un problema fondamentale di ottimizzazione: trasportare fluidi (sangue, linfa) in modo efficiente attraverso reti ramificate. Il fisiologo Cecil Murray, nel 1926, derivò una relazione matematica che governa il diametro ottimale dei vasi sanguigni.

Il problema che Murray considerò è il seguente: il sangue deve essere trasportato attraverso vasi di vario diametro. Vasi più larghi richiedono più materiale biologico per essere costruiti e mantenuti (costo metabolico), ma offrono meno resistenza al flusso (richiedono meno energia per pompare il sangue). Vasi più stretti risparmiano materiale ma aumentano la resistenza al flusso.

Per un fluido viscoso che scorre in un tubo cilindrico, la legge di Poiseuille descrive la relazione tra portata Q , caduta di pressione ΔP , raggio r , lunghezza l e viscosità η :

$$Q = (\pi r^4 \Delta P) / (8 \eta l)$$

La resistenza al flusso è definita come $R = \Delta P / Q$, quindi:

$$R = (8 \eta l) / (\pi r^4)$$

Notiamo che la resistenza è inversamente proporzionale alla quarta potenza del raggio. Questo significa che raddoppiare il raggio riduce la resistenza di un fattore 16.

La potenza necessaria per mantenere il flusso contro questa resistenza è:

$$P_{\text{flusso}} = Q \times \Delta P = Q^2 R = Q^2 \times (8 \eta l) / (\pi r^4)$$

Il costo metabolico per mantenere il volume del vaso è proporzionale al suo volume:

$$C_{\text{volume}} = k \times \pi r^2 l$$

dove k è una costante che rappresenta il costo metabolico per unità di volume di tessuto vascolare.

Il costo totale è la somma della potenza richiesta per il flusso (che deve essere fornita continuamente dal cuore) e del costo di mantenimento:

$$C_{\text{totale}} = Q^2 \times (8 \eta l) / (\pi r^4) + k \times \pi r^2 l$$

Per trovare il raggio ottimale che minimizza questo costo, deriviamo rispetto a r e poniamo uguale a zero:

$$dC/dr = -4Q^2 \times (8 \eta l) / (\pi r^5) + 2k \times \pi r l = 0$$

Risolviendo per r :

$$r^3 = (16 \eta Q^2) / (\pi^2 k)$$

Questa equazione ci dice che il raggio ottimale è proporzionale a $Q^{2/3}$. In altre parole:

$$r \propto Q^{1/3}$$

Poiché la portata Q è proporzionale all'area della sezione trasversale per la velocità media, e vogliamo sapere come i raggi si relazionano quando un vaso si biforca, consideriamo un vaso madre di raggio r_0 che si divide in due vasi figlie di raggi r_1 e r_2 .

La conservazione del flusso richiede: $Q_0 = Q_1 + Q_2$

Applicando la relazione $r \propto Q^{1/3}$:

$$r_0^3 = r_1^3 + r_2^3$$

Questa è la **legge di Murray**, e afferma che al punto di biforcazione, il cubo del raggio del vaso madre uguaglia la somma dei cubi dei raggi dei vasi figlie.

6.2 Verifica Sperimentale della Legge di Murray

La legge di Murray è stata verificata sperimentalmente in numerosi sistemi biologici. Misurazioni accurate dei vasi sanguigni in vari animali mostrano che la relazione $r_0^3 = r_1^3 + r_2^3$ è soddisfatta con notevole precisione.

Ad esempio, nelle arterie coronarie umane, studi hanno misurato i diametri ai punti di biforcazione e trovato che l'esponente nella relazione $r_0^n = r_1^n + r_2^n$ è tipicamente tra 2,7 e 3,1, molto vicino al valore teorico di 3.

Anche in sistemi vascolari di piante si osserva una relazione simile, sebbene l'esponente possa differire leggermente (tipicamente tra 2,0 e 2,5) a causa delle diverse proprietà del fluido trasportato (linfa invece di sangue) e della diversa struttura dei tessuti.

È importante notare che la legge di Murray è una conseguenza dell'ottimizzazione energetica, non una legge fisica fondamentale. Deviazioni dalla legge si osservano in situazioni patologiche (ad esempio, vasi danneggiati o malformati) o quando altri vincoli diventano dominanti.

6.3 Venature delle Foglie: Reti di Trasporto Ottimali

Le venature nelle foglie formano reti intricate che trasportano acqua e nutrienti. Queste reti devono soddisfare vincoli multipli:

1. Fornire acqua e nutrienti a ogni parte della foglia
2. Minimizzare la quantità di tessuto vascolare (che è costoso da costruire e mantenere)
3. Essere robusto contro danni locali
4. Minimizzare la distanza totale di trasporto

Questo è un problema di ottimizzazione di rete complesso, analogo ai problemi affrontati nell'ingegneria delle reti di distribuzione (acqua, elettricità, telecomunicazioni).

La struttura tipica delle venature segue un pattern gerarchico:

- Una vena principale (nervatura centrale) attraversa la foglia
- Vene secondarie si diramano dalla principale
- Vene terziarie formano una rete interconnessa più fine
- Vene di ordine superiore creano una mesh sempre più fitta

Questo pattern gerarchico emerge naturalmente come soluzione a un problema di ottimizzazione. Marcelo Magnasco e collaboratori hanno dimostrato che le venature delle foglie possono essere previste risolvendo un problema di flusso ottimale su una rete.

Il modello considera la foglia come una griglia di punti, ognuno dei quali richiede una certa quantità di fluido (domanda). Il problema è costruire una rete di condotti che connetta tutti i punti a una sorgente centrale, minimizzando il costo totale:

$$C = \sum_{\text{archi}} c(Q_i, l_i)$$

dove la somma è su tutti gli archi della rete, Q_i è il flusso attraverso l'arco i , l_i è la sua lunghezza, e c è una funzione di costo.

Se assumiamo che il costo per un arco sia:

$$c(Q, l) = a l + b(Q/r^4)l$$

dove il primo termine rappresenta il costo di costruzione (proporzionale alla lunghezza) e il secondo la resistenza al flusso (dalla legge di Poiseuille), e ottimizziamo simultaneamente la topologia della rete e i raggi dei condotti, emerge naturalmente una struttura gerarchica simile a quella osservata nelle foglie reali.

6.4 Reti di Drenaggio e Fiumi: Analogie Geomorfologiche

Le reti fluviali mostrano notevoli similarità con le reti vascolari biologiche, suggerendo che principi di ottimizzazione simili operano in contesti molto diversi.

Un bacino fluviale drena l'acqua piovana attraverso una rete gerarchica di corsi d'acqua: piccoli rigagnoli si uniscono in ruscelli, ruscelli in torrenti, torrenti in fiumi, fino al fiume principale che sfocia in mare.

Questa struttura può essere compresa come la soluzione a un problema di ottimizzazione: minimizzare l'energia totale dissipata nel trasporto dell'acqua dal bacino al mare.

La legge di Horton, scoperta empiricamente dal geomorfologo Robert Horton nel 1945, descrive proprietà statistiche delle reti fluviali:

1. **Legge dei numeri:** Il rapporto tra il numero di corsi d'acqua di ordine n e quelli di ordine $n+1$ è approssimativamente costante (rapporto di biforcazione)
2. **Legge delle lunghezze:** Il rapporto tra la lunghezza media di corsi d'acqua di ordine $n+1$ e quelli di ordine n è costante
3. **Legge delle aree:** Il rapporto tra l'area media drenata da corsi d'acqua di ordine $n+1$ e n è costante

Questi rapporti sono notevolmente simili tra bacini fluviali diversi in tutto il mondo, suggerendo principi universali.

Modelli basati sull'ottimizzazione energetica riproducono queste leggi. Per esempio, il modello di optimal channel networks (OCN) genera reti fluviali sintetiche minimizzando l'energia totale dissipata, definita come:

$$E = \sum_{\text{links}} Q_i^\gamma L_i$$

dove Q_i è la portata nel link i , L_i è la sua lunghezza, e γ è un esponente (tipicamente $\gamma \approx 0,5$ basato su considerazioni idrauliche). Le reti generate da questo principio di ottimizzazione mostrano proprietà statistiche molto simili a quelle osservate in natura.

6.5 Ottimizzazione Multi-Obiettivo nelle Reti Biologiche

Le reti biologiche reali devono bilanciare obiettivi multipli, spesso in conflitto:

Efficienza vs. Robustezza: Una rete ottimizzata puramente per l'efficienza del trasporto potrebbe essere vulnerabile a danni locali. Una rete altamente interconnessa con percorsi ridondanti è più robusta ma richiede più materiale.

Costo di costruzione vs. Costo operativo: Come abbiamo visto nella legge di Murray, c'è un trade-off tra il costo di mantenere tessuto vascolare e il costo energetico di pompare fluidi attraverso vasi stretti.

Velocità vs. Precisione: In alcuni sistemi (come il sistema nervoso), c'è un trade-off tra la velocità di trasmissione del segnale (favorita da assoni spessi e mielinizzati) e il costo metabolico.

Matematicamente, i problemi di ottimizzazione multi-obiettivo non hanno una singola soluzione ottimale, ma un insieme di soluzioni **Pareto-ottimali**: configurazioni tali che non è possibile migliorare un obiettivo senza peggiorare almeno uno degli altri.

L'insieme delle soluzioni Pareto-ottimali forma il **fronte di Pareto** nello spazio degli obiettivi. Le configurazioni osservate in natura tipicamente si trovano su o vicino a questo fronte, riflettendo i diversi compromessi che diverse specie o diversi organi hanno evoluto in risposta a pressioni selettive specifiche.

Capitolo 7: Forme Biologiche e Principi Fisici

7.1 La Crescita di D'Arcy Thompson

Nel suo monumentale lavoro "On Growth and Form" (1917), il biologo e matematico scozzese D'Arcy Wentworth Thompson propose che molte forme biologiche possono essere comprese come conseguenze di principi fisici e matematici piuttosto che solo di contingenze evolutive.

Thompson osservò che forme simili appaiono in contesti molto diversi quando i vincoli fisici sono simili. Per esempio, la forma di una medusa sospesa nell'acqua assomiglia a quella di una goccia di fluido viscoso, entrambe governate dalle stesse equazioni di equilibrio tra tensione superficiale, gravità e forze viscosi.

Un concetto chiave nel lavoro di Thompson è quello delle **trasformazioni coordinate**: molte forme biologiche correlate evolutivamente possono essere trasformate l'una nell'altra attraverso deformazioni matematiche semplici del sistema di coordinate. Per esempio, i crani di diverse specie di primati possono essere messi in corrispondenza attraverso trasformazioni che allungano o comprimono selettivamente diverse regioni.

Questo suggerisce che l'evoluzione spesso opera modificando parametri di crescita differenziale piuttosto che reinventando completamente le forme. La matematica sottostante rimane la stessa; cambiano solo i parametri.

7.2 Leggi di Scala: Rapporti Allometrici

Le dimensioni e le proporzioni degli organismi non scalano arbitrariamente. Esistono relazioni matematiche precise, chiamate leggi allometriche, che descrivono come diverse caratteristiche biologiche scalano con la dimensione corporea.

La forma generale di una legge allometrica è:

$$Y = aM^b$$

dove Y è la caratteristica di interesse (ad esempio, tasso metabolico, massa del cervello, lunghezza delle ossa), M è la massa corporea, a è una costante di proporzionalità, e b è l'esponente allometrico.

L'esponente b rivela come la caratteristica scala:

- $b = 1$: scala isometricamente (proporzionalmente alla massa)
- $b < 1$: scala con allometria negativa (cresce meno rapidamente della massa)
- $b > 1$: scala con allometria positiva (cresce più rapidamente della massa)

Legge di Kleiber (metabolismo): Il tasso metabolico basale scala come $M^{3/4}$:

$$B = B_0 M^{3/4}$$

Questo significa che animali più grandi hanno un tasso metabolico per unità di massa più basso. Un elefante consuma meno energia per chilogrammo di corpo rispetto a un topo.

Legge delle superfici (resistenza ossea): La resistenza delle ossa deve supportare il peso dell'animale, che scala come M. Ma la resistenza di una colonna è proporzionale alla sua area della sezione trasversale, che scala come L^2 (dove L è una lunghezza caratteristica). Poiché il volume (e quindi la massa) scala come L^3 , abbiamo:

$$M \propto L^3, \text{ quindi } L \propto M^{1/3}$$

L'area della sezione trasversale delle ossa deve quindi scalare come: $A \propto L^2 \propto M^{2/3}$

Questo è stato verificato empiricamente: il diametro delle ossa degli arti scala approssimativamente come $M^{(1/3)}$ nei mammiferi.

7.3 La Teoria Metabolica dell'Ecologia

Geoffrey West, James Brown e Brian Enquist hanno sviluppato una teoria unificata che spiega molte leggi allometriche come conseguenze dell'ottimizzazione delle reti di distribuzione nei sistemi biologici.

La loro teoria si basa su tre assunzioni:

1. Le reti biologiche (vascolari, respiratorie) riempiono lo spazio tridimensionale dell'organismo (sono **space-filling**)
2. Le unità terminali della rete (capillari, alveoli) sono invarianti rispetto alla dimensione dell'organismo
3. La rete è ottimizzata per minimizzare l'energia di distribuzione

Da queste assunzioni, derivano matematicamente che molte caratteristiche biologiche dovrebbero scalare con esponenti che sono multipli di $1/4$:

- Tasso metabolico: $M^{(3/4)}$
- Frequenza cardiaca: $M^{(-1/4)}$
- Durata della vita: $M^{(1/4)}$
- Raggio dell'aorta: $M^{(3/8)}$
- Densità capillare: $M^{(-1/4)}$

La derivazione completa è matematicamente sofisticata, ma l'idea chiave è che una rete tridimensionale che deve servire l'intero volume dell'organismo (che scala come M) attraverso unità terminali di dimensione fissa introduce automaticamente la scala di lunghezza $M^{(1/4)}$ che appare negli esponenti.

L'argomento procede così: consideriamo una rete gerarchica con N livelli di ramificazione. Al livello k , ci sono n_k vasi di raggio r_k e lunghezza l_k . Le condizioni di ottimizzazione (simili alla legge di Murray) impongono:

$n_k r_k^2 l_k \approx \text{costante}$ (conservazione del volume vascolare) $n_k (r_k)^3 \approx \text{costante}$ (legge di Murray al punto di ramificazione)

Combinando queste relazioni e imponendo che la rete riempia lo spazio tridimensionale, si ottiene che le caratteristiche vascolari devono scalare con esponenti frazionari che sono multipli di $1/4$.

7.4 Forme Ottimali sotto Gravità: Il Problema della Torre

Un problema classico di ottimizzazione strutturale è quello della **torre di massima altezza**: qual è la forma di una torre autoportante (che deve supportare solo il proprio peso) di altezza massima, data una quantità fissa di materiale?

Questo problema fu considerato da Galileo nel suo "Discorsi e dimostrazioni matematiche" (1638). Galileo osservò che esiste un limite superiore all'altezza di qualsiasi struttura fatta di un materiale dato, e che questo limite dipende dalle proprietà del materiale.

Formuliamo il problema matematicamente. Consideriamo una torre verticale di altezza h e area della sezione trasversale $A(z)$, dove z è la coordinata verticale misurata dalla base. Il materiale ha densità ρ e resistenza alla compressione $\sigma_{\max}$.

Lo stress (forza per unità di area) al livello z è dovuto al peso di tutto il materiale sopra z :

$$\sigma(z) = (\rho g / A(z)) \int_z^h A(\xi) d\xi$$

dove g è l'accelerazione gravitazionale. Perché la torre non collassi, dobbiamo avere $\sigma(z) \leq \sigma_{\max}$ per ogni z .

Per massimizzare l'altezza data una quantità fissa di materiale V , dobbiamo ottimizzare la funzione $A(z)$. Usando il calcolo variazionale con il vincolo:

$$\int_0^h A(z) dz = V$$

si trova che la forma ottimale è esponenziale:

$$A(z) = A_0 e^{-(\rho g z / \sigma_{\max})}$$

dove A_0 è l'area alla base. L'altezza massima è ottenuta quando $\sigma(0) = \sigma_{\max}$, il che dà:

$$h_{\max} = (\sigma_{\max}) / (\rho g)$$

Questa è l'**altezza limite di Galileo**: indipendentemente dalla forma, nessuna torre fatta di un certo materiale può superare questa altezza critica.

Per il granito ($\sigma_{\max} \approx 200$ MPa, $\rho \approx 2700$ kg/m³): $h_{\max} \approx 7,5$ km

Per l'acciaio ($\sigma_{\max} \approx 400$ MPa, $\rho \approx 7800$ kg/m³): $h_{\max} \approx 5,2$ km

Questi calcoli spiegano perché le montagne terrestri non superano circa 10 km di altezza e perché gli alberi più alti (sequoie costiere) raggiungono circa 115 m, ben al di sotto del limite teorico per il legno, ma limitati invece dalla capacità di trasportare acqua contro la gravità.

7.5 Il Numero di Reynolds e l'Ottimizzazione del Nuoto

Gli organismi acquatici affrontano vincoli fisici determinati dal numero di Reynolds, un parametro adimensionale che caratterizza il flusso di fluidi:

$$Re = (\rho v L) / \eta$$

dove ρ è la densità del fluido, v è la velocità caratteristica, L è una lunghezza caratteristica dell'organismo, e η è la viscosità dinamica del fluido.

Per $Re \ll 1$ (flusso di Stokes, regime viscoso dominante), le forze inerziali sono trascurabili rispetto alle forze viscosi. Questo è il regime dei microrganismi come batteri e protozoi.

Per $Re \gg 1$ (flusso turbolento, regime inerziale), le forze inerziali dominano. Questo è il regime dei pesci grandi e dei mammiferi marini.

Le strategie di locomozione ottimali sono radicalmente diverse nei due regimi.

Regime a basso Reynolds ($Re < 1$)

Nel regime di Stokes, il flusso è reversibile: se un organismo esegue un movimento e poi lo inverte esattamente, torna alla posizione iniziale senza spostamento netto. Questo è il **teorema dello scallop** di Purcell: un organismo simmetrico che si apre e si chiude (come una conchiglia) non può nuotare nel regime di Stokes.

Per muoversi, i microrganismi devono rompere la simmetria temporale. I batteri usano flagelli elicoidali rotanti; i protozoi usano ciglia che eseguono movimenti asimmetrici (colpo di potenza seguito da recupero con geometria diversa).

L'efficienza della propulsione nel regime di Stokes può essere quantificata. Per un flagello elicoidale rotante, il rapporto tra la velocità di avanzamento v e la velocità di rotazione ω è:

$$v/\omega = (L \tan \alpha)/(4\pi) \times (1 - c_N/c_T)$$

dove L è la lunghezza del flagello, α è l'angolo dell'elica, e c_N/c_T è il rapporto tra i coefficienti di resistenza normale e tangenziale. L'ottimizzazione di α massimizza l'efficienza propulsiva, portando a valori ottimali di $\alpha \approx 30-40^\circ$, consistenti con osservazioni su batteri reali.

Regime ad alto Reynolds ($Re > 1000$)

Nei pesci e mammiferi marini, la resistenza idrodinamica è dominata dalla drag pressure. La forma del corpo è ottimizzata per minimizzare questa resistenza.

La forma ottimale è il **corpo di minima resistenza**, studiato da von Kármán e altri aerodinamici. Per un corpo di rivoluzione di lunghezza L e diametro massimo d , il coefficiente di resistenza c_D dipende dalla forma longitudinale.

Curiosamente, molti pesci veloci (tonni, squali) hanno forme che approssimano il profilo ottimale teorico con notevole precisione. Il rapporto lunghezza/diametro è tipicamente 4-6, vicino al valore ottimale teorico di circa 4,5 per minimizzare la resistenza.

La forma ottimale può essere trovata risolvendo le equazioni di Navier-Stokes con un vincolo di volume fisso e minimizzando:

$$D = \iint \tau \cdot n \, dS$$

dove l'integrale è sulla superficie del corpo, τ è lo stress viscoso, e n è la normale alla superficie.

7.6 Ottimizzazione Termica: Forma e Dimensione degli Animali

La termoregolazione impone vincoli importanti sulla forma e dimensione degli animali. Gli organismi endotermici (a sangue caldo) devono bilanciare la produzione di calore metabolico con la perdita di calore attraverso la superficie.

La produzione di calore è proporzionale alla massa (più precisamente, a $M^{3/4}$) secondo la legge di Kleiber): $H_{\text{prod}} \propto M^{3/4}$

La perdita di calore è proporzionale alla superficie: $H_{\text{loss}} \propto S \propto M^{2/3}$

(assumendo forme geometricamente simili, dove $S \propto L^2$ e $M \propto L^3$).

Il rapporto produzione/perdita scala come: $H_{\text{prod}}/H_{\text{loss}} \propto M^{3/4}/M^{2/3} = M^{1/12}$

Questo significa che animali più grandi hanno relativamente meno difficoltà a mantenere la temperatura corporea in ambienti freddi, mentre animali piccoli perdono calore rapidamente.

Questo spiega la **regola di Bergmann**: all'interno di una specie, gli individui in climi freddi tendono a essere più grandi di quelli in climi caldi. Per esempio, gli orsi polari sono considerevolmente più grandi degli orsi bruni in regioni temperate.

La **regola di Allen** complementa quella di Bergmann: le appendici (orecchie, code, arti) tendono a essere più corte in climi freddi. Questo riduce la superficie esposta e minimizza la perdita di calore. Confronta le orecchie piccole di una volpe artica con quelle enormi di un fennec del deserto.

Matematicamente, possiamo formulare questo come un problema di ottimizzazione. Per un animale di massa M in un ambiente a temperatura T_{amb} , la potenza termica netta è:

$$P_{\text{net}} = aM^{3/4} - h(T_{\text{body}} - T_{\text{amb}})S$$

dove a è una costante metabolica, h è il coefficiente di trasferimento termico, T_{body} è la temperatura corporea, e S è la superficie.

Per mantenere l'omeostasi termica, $P_{\text{net}} = 0$, il che determina la relazione tra M e S (e quindi la forma) per un dato ambiente termico.

7.7 Packing Ottimale: Semi, Cellule ed Embrioni

Il problema di impaccare oggetti in modo efficiente nello spazio disponibile appare in molti contesti biologici.

Semi in un frutto: I semi devono essere disposti in modo da massimizzare il numero che può essere contenuto nel volume del frutto. Per semi sferici, il packing ottimale è il **face-centered cubic (FCC)** o **hexagonal close packing (HCP)**, entrambi con densità di packing:

$$\eta = \pi/(3\sqrt{2}) \approx 0,7405$$

Questo significa che circa il 74% dello spazio è occupato dalle sfere, il resto è spazio vuoto. Johannes Kepler congetturò nel 1611 che questo è il packing più denso possibile per sfere.

identiche; la congettura fu dimostrata (con metodi assistiti da computer) da Thomas Hales nel 1998.

Molti frutti mostrano disposizioni di semi vicine all'ottimale. Il melograno, per esempio, ha semi quasi sferici impaccati con densità molto alta, sebbene irregolarità locali portino a deviazioni dal packing perfettamente regolare.

Cellule in tessuti: Le cellule in tessuti epiteliali tendono ad assumere forme che minimizzano la tensione superficiale totale delle membrane cellulari. Per un tessuto bidimensionale, questo porta a configurazioni dove le cellule sono approssimativamente esagonali, simile al problema del favo.

In tre dimensioni, il poliedro che minimizza la superficie a volume costante e che può riempire lo spazio è il **tetrakaidecahedro di Kelvin** (14 facce: 6 quadrate e 8 esagonali). Lord Kelvin propose nel 1887 che questa fosse la struttura ottimale per una schiuma.

Tuttavia, nel 1993, i fisici Denis Weaire e Robert Phelan scoprirono una struttura ancora migliore (con superficie leggermente minore): la **struttura di Weaire-Phelan**, utilizzata nella progettazione del National Aquatics Centre di Pechino per le Olimpiadi 2008.

Alcune cellule biologiche e strutture di schiume naturali approssimano queste geometrie ottimali, sebbene vincoli biologici aggiuntivi (come polarità cellulare, adesione differenziale, forze citoscheletriche) possano portare a deviazioni.

Conclusione della Parte II

Abbiamo visto come i principi di ottimizzazione permeano le strutture e i processi biologici a ogni scala. Dalle celle esagonali degli alveari alle spirali auree della filotassi, dalle ragnatele ottimizzate alle reti vascolari che seguono la legge di Murray, la natura risolve continuamente problemi complessi di ottimizzazione.

Questi non sono semplici coincidenze o analogie superficiali. La matematica del calcolo variazionale e dell'ottimizzazione descrive realmente i vincoli fisici e i trade-off energetici che gli organismi affrontano. La selezione naturale favorisce le configurazioni che meglio risolvono questi problemi di ottimizzazione, portando alla convergenza evolutiva verso soluzioni matematicamente ottimali.

È importante riconoscere che l'ottimizzazione in natura è raramente assoluta. Vincoli multipli, spesso in conflitto, devono essere bilanciati. Il risultato è tipicamente una soluzione di compromesso sul fronte di Pareto, ottimale rispetto al particolare insieme di vincoli e costi che l'organismo affronta nel suo ambiente specifico.

Inoltre, la storia evolutiva impone vincoli: l'ottimizzazione opera su strutture esistenti, non può ricominciare da zero. Questo porta a soluzioni "sufficientemente buone" piuttosto che perfettamente ottimali - quello che il biologo François Jacob chiamò "bricolage" evoluzionistico.

Ciononostante, la prevalenza e la precisione con cui principi matematici di ottimizzazione si manifestano in natura rimane uno dei fenomeni più notevoli e belli della biologia.

Comprendere questi principi non solo approfondisce la nostra comprensione del mondo naturale, ma fornisce anche ispirazione per soluzioni tecnologiche, come vedremo nella prossima parte di questo trattato.

PARTE III: ALGORITMI BIO-ISPIRATI

Capitolo 8: Ant Colony Optimization

8.1 Come le Formiche Trovano il Percorso Ottimale

Le formiche, individualmente semplici creature con capacità cognitive limitate, collettivamente risolvono problemi complessi di ottimizzazione. Il caso più studiato è la loro capacità di trovare il percorso più breve tra il nido e una fonte di cibo, un problema noto in informatica come il Problema del Commesso Viaggiatore (Traveling Salesman Problem, TSP).

Il meccanismo alla base di questa capacità è la **stigmergia**, una forma di comunicazione indiretta attraverso modifiche dell'ambiente. Le formiche depositano feromoni lungo i percorsi che seguono. I feromoni evaporano col tempo, ma percorsi più brevi (percorsi più veloci) ricevono rinforzo di feromoni più frequentemente, creando un feedback positivo che guida la colonia verso la soluzione ottimale.

Osserviamo il processo in dettaglio. Inizialmente, le formiche esplorano casualmente. Supponiamo che esistano due percorsi tra nido e cibo: uno lungo L_1 e uno lungo L_2 , con $L_1 < L_2$. All'inizio, assumiamo che un numero uguale di formiche scelga ciascun percorso casualmente.

Una formica che percorre il percorso più corto impiega tempo $t_1 = L_1/v$ (dove v è la velocità), mentre una sul percorso lungo impiega $t_2 = L_2/v$. Nel tempo t_1 , le formiche sul percorso corto completano un viaggio di andata e ritorno, depositando feromoni due volte. Le formiche sul percorso lungo sono ancora in viaggio.

Quando nuove formiche devono scegliere un percorso, trovano più feromoni sul percorso corto e sono quindi più propense a sceglierlo. Questo rinforza ulteriormente il percorso corto, creando un ciclo di feedback positivo che rapidamente concentra la maggior parte del traffico sul percorso ottimale.

8.2 Il Modello Matematico

Formalizziamo questo comportamento. Sia $\tau_{ij}(t)$ la quantità di feromone sull'arco che connette il nodo i al nodo j al tempo t . La probabilità che una formica in i scelga di muoversi verso j è:

$$p_{ij} = [\tau_{ij}^\alpha \cdot \eta_{ij}^\beta] / \sum_k [\tau_{ik}^\alpha \cdot \eta_{ik}^\beta]$$

dove:

- η_{ij} è la "visibilità" dell'arco (tipicamente $\eta_{ij} = 1/d_{ij}$, dove d_{ij} è la distanza)
- α controlla l'importanza relativa del feromone
- β controlla l'importanza relativa della visibilità
- la somma è su tutti i nodi k adiacenti a i

L'aggiornamento del feromone avviene in due fasi:

Evaporazione: $\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t)$

dove $\rho \in (0,1)$ è il tasso di evaporazione.

Deposizione: $\tau_{ij}(t+1) = \tau_{ij}(t+1) + \sum_m \Delta\tau_{ij}^m$

dove la somma è su tutte le formiche m che hanno attraversato l'arco (i,j) , e:

$$\Delta\tau_{ij}^m = Q/L_m$$

dove Q è una costante e L_m è la lunghezza del tour completato dalla formica m .

Questo modello cattura l'essenza del comportamento delle formiche reali: percorsi più corti ricevono più feromone per unità di tempo, l'evaporazione previene la convergenza prematura su soluzioni sub-ottimali, e la componente stocastica permette l'esplorazione di nuove soluzioni.

8.3 Algoritmo Ant Colony Optimization per il TSP

L'Ant Colony Optimization (ACO) traduce questo comportamento naturale in un algoritmo computazionale. Per il Problema del Commesso Viaggiatore, dove dobbiamo visitare n città minimizzando la distanza totale, l'algoritmo procede così:

Inizializzazione:

- Posiziona m formiche casualmente sulle n città
- Inizializza il feromone: $\tau_{ij} = \tau_0$ per tutti gli archi
- Imposta i parametri: α , β , ρ , Q

Iterazione (ripeti fino a convergenza):

1. Ogni formica costruisce un tour:
 - Partendo dalla sua città iniziale
 - Scegli iterativamente la prossima città non visitata secondo p_{ij}
 - Continua finché tutte le città sono visitate
 - Torna alla città iniziale
2. Aggiorna i feromoni:
 - Applica evaporazione a tutti gli archi
 - Ogni formica deposita feromone sul suo tour
3. (Opzionale) Elitismo: rinforza ulteriormente il miglior tour trovato finora

Terminazione: L'algoritmo termina dopo un numero fisso di iterazioni o quando la soluzione non migliora per diverse iterazioni consecutive.

8.4 Varianti e Miglioramenti

L'algoritmo ACO base può essere migliorato in diversi modi:

Ant System (AS): La versione originale di Dorigo (1992), dove tutte le formiche depositano feromone proporzionalmente alla qualità del loro tour.

Elitist Ant System: Solo le migliori formiche (o solo la migliore) depositano feromone, accelerando la convergenza.

Rank-Based Ant System: Le formiche sono ordinate per qualità dei tour, e il feromone depositato è pesato in base al rank.

Max-Min Ant System (MMAS): Introduce limiti superiori e inferiori sui livelli di feromone:
 $\tau_{\min} \leq \tau_i \leq \tau_{\max}$

Questo previene che alcuni archi accumulino troppo feromone (stagnazione) o perdano tutto il feromone (perdita di informazione).

Ant Colony System (ACS): Introduce:

1. Una regola di transizione pseudo-casuale che bilancia exploitation ed exploration
2. Aggiornamento locale del feromone durante la costruzione del tour
3. Aggiornamento globale solo sul miglior tour

8.5 Analisi della Convergenza

La convergenza di ACO può essere analizzata formalmente. La quantità chiave è la **probabilità di convergenza**: la probabilità che l'algoritmo converga alla soluzione ottimale in tempo finito.

Thomas Stützle e Holger Hoos hanno dimostrato che sotto certe condizioni (principalmente legate ai parametri α , β , ρ), ACO converge con probabilità 1 alla soluzione ottimale, sebbene il tempo di convergenza possa essere esponenziale nel caso peggiore.

La convergenza può essere compresa in termini di catene di Markov. Lo stato del sistema è la distribuzione dei feromoni. Le transizioni di stato sono determinate dai tour costruiti dalle formiche e dall'aggiornamento dei feromoni. Sotto condizioni appropriate, questa catena di Markov è ergodica e converge a una distribuzione stazionaria che concentra la probabilità sulla soluzione ottimale.

8.6 Applicazioni oltre il TSP

ACO è stato applicato con successo a molti altri problemi di ottimizzazione combinatoria:

Routing in reti di telecomunicazioni: Le "formiche" esplorano percorsi attraverso la rete, depositando feromoni sui link. Il traffico reale è poi instradato preferenzialmente su link con alto feromone, adattandosi dinamicamente a congestioni e guasti.

Scheduling: Assegnare task a risorse minimizzando il tempo totale o il costo. Le formiche costruiscono schedule validi e il feromone guida verso schedule ottimali.

Protein folding: Trovare la configurazione tridimensionale di minima energia di una proteina. Le formiche esplorano lo spazio conformazionale, con il feromone che rinforza configurazioni a bassa energia.

Vehicle routing: Generalizzazione del TSP dove multipli veicoli devono servire clienti, rispettando vincoli di capacità e finestre temporali.

In tutti questi casi, la forza di ACO risiede nella sua capacità di bilanciare exploration (esplorazione di nuove soluzioni) ed exploitation (raffinamento di buone soluzioni conosciute), adattandosi dinamicamente attraverso il feedback del feromone.

Capitolo 9: Particle Swarm Optimization

9.1 Il Comportamento Collettivo di Stormi e Banchi

Gli stormi di uccelli e i banchi di pesci mostrano comportamenti collettivi coordinati notevoli, emergenti da regole individuali semplici. Ogni individuo ajusta la propria velocità e direzione basandosi su tre principi:

1. **Separazione:** Evita collisioni con vicini troppo prossimi
2. **Allineamento:** Muoviti nella direzione media dei vicini
3. **Coesione:** Muoviti verso la posizione media dei vicini

Queste regole, formulate originariamente da Craig Reynolds nel 1986 nel suo modello "boids", producono il movimento fluido e organizzato che osserviamo negli stormi reali.

Il biologo James Kennedy e l'ingegnere Russell Eberhart, nel 1995, si resero conto che questo comportamento poteva essere adattato per risolvere problemi di ottimizzazione. L'idea è che ogni "particella" in uno sciame rappresenti una soluzione candidata in uno spazio di ricerca. Le particelle si muovono attraverso lo spazio, influenzate dalla loro esperienza personale (migliore posizione che hanno visitato) e dall'esperienza collettiva (migliore posizione trovata da tutto lo sciame).

9.2 Il Modello Matematico di PSO

Consideriamo un problema di ottimizzazione: minimizzare una funzione $f(x)$ dove $x \in \mathbb{R}^n$. Abbiamo N particelle, ciascuna caratterizzata da:

- Posizione corrente: x_i
- Velocità corrente: v_i
- Migliore posizione personale: p_i (personal best)

Inoltre, tracciamo la migliore posizione trovata da tutto lo sciame: g (global best).

Ad ogni iterazione, la velocità e la posizione di ogni particella vengono aggiornate secondo:

$$v_i(t+1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot (p_i - x_i(t)) + c_2 \cdot r_2 \cdot (g - x_i(t))$$

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

dove:

- w è il coefficiente di inerzia (tipicamente 0,4 - 0,9)
- c_1 è il coefficiente cognitivo (attrazione verso il personal best, tipicamente ~2)
- c_2 è il coefficiente sociale (attrazione verso il global best, tipicamente ~2)
- r_1, r_2 sono numeri casuali uniformi in $[0,1]$, rigenerati ad ogni iterazione

Il termine $w \cdot v_i$ rappresenta l'inerzia: la particella tende a continuare nella direzione che stava seguendo. Il termine $c_1 \cdot r_1 \cdot (p_i - x_i)$ attira la particella verso la sua migliore posizione personale. Il termine $c_2 \cdot r_2 \cdot (g - x_i)$ attrae verso la migliore posizione globale.

La componente stocastica (r_1 e r_2) introduce variabilità che aiuta a evitare minimi locali e esplora lo spazio di ricerca in modo più completo.

9.4 Analisi del Comportamento di PSO

Il comportamento di PSO può essere analizzato studiando la dinamica delle particelle. Consideriamo una singola particella in una dimensione, con il global best fisso in posizione g e il personal best in p .

L'equazione di aggiornamento diventa:

$$v(t+1) = w \cdot v(t) + c_1 \cdot r_1 \cdot (p - x(t)) + c_2 \cdot r_2 \cdot (g - x(t))$$

Questa è un'equazione alle differenze stocastica. Il comportamento medio può essere analizzato prendendo il valore atteso e trattando r_1 e r_2 come costanti pari ai loro valori medi (0,5).

Definendo $\phi_1 = c_1/2$ e $\phi_2 = c_2/2$, e ponendo $\phi = \phi_1 + \phi_2$, possiamo riscrivere:

$$v(t+1) = w \cdot v(t) + \phi_1 \cdot (p - x(t)) + \phi_2 \cdot (g - x(t))$$

La convergenza dipende criticamente dai parametri w , c_1 e c_2 . Maurice Clerc e James Kennedy hanno derivato la "constriction factor" formula per garantire convergenza:

$$v(t+1) = \chi[v(t) + c_1 \cdot r_1 \cdot (p - x(t)) + c_2 \cdot r_2 \cdot (g - x(t))]$$

$$\text{dove: } \chi = 2 / (2 - \phi - \sqrt{\phi^2 - 4\phi})$$

con $\phi = c_1 + c_2 > 4$.

Tipicamente si sceglie $c_1 = c_2 = 2,05$, dando $\chi \approx 0,7298$.

9.5 Varianti di PSO

Topology delle connessioni: Nella versione base (gbest), tutte le particelle sono influenzate dal migliore globale. Alternative includono:

- **lbest:** Ogni particella è influenzata solo dai migliori in un vicinato locale
- **Ring topology:** Le particelle formano un anello, ogni particella comunica con i vicini immediati
- **Star topology:** Una particella centrale comunica con tutte le altre

PSO binario: Per problemi di ottimizzazione combinatoria, le posizioni sono vettori binari e le velocità rappresentano probabilità di flip dei bit.

Multi-objective PSO: Estensioni per ottimizzare simultaneamente obiettivi multipli, mantenendo un set di soluzioni Pareto-ottimali.

9.6 Applicazioni di PSO

PSO è particolarmente efficace per:

Addestramento di reti neurali: Ottimizzare i pesi della rete come alternativa alla backpropagation.

Tuning di iperparametri: Trovare i migliori iperparametri per algoritmi di machine learning.

Progettazione ingegneristica: Ottimizzare forme di componenti strutturali, circuiti elettronici, antenne.

Scheduling e resource allocation: Problemi dove lo spazio di ricerca è continuo o misto.

Capitolo 10: Algoritmi Genetici

10.1 I Principi dell'Evoluzione Naturale

L'evoluzione biologica opera attraverso meccanismi semplici ma potenti:

1. **Variazione:** Gli individui differiscono geneticamente
2. **Selezione:** Individui più adatti hanno maggiore probabilità di riprodursi
3. **Eredità:** I tratti vengono trasmessi alla progenie
4. **Ricombinazione:** Il materiale genetico dei genitori si combina

Gli algoritmi genetici (Genetic Algorithms, GA), sviluppati principalmente da John Holland negli anni '70, traducono questi principi in un metodo di ottimizzazione computazionale.

10.2 Struttura di un Algoritmo Genetico

Un GA opera su una **popolazione** di **individui** (soluzioni candidate). Ogni individuo è rappresentato da un **cromosoma** (tipicamente una stringa di bit, ma possono essere usate altre rappresentazioni).

Il ciclo base è:

1. **Inizializzazione**: Genera popolazione iniziale casuale
2. **Valutazione**: Calcola il fitness di ogni individuo
3. **Selezione**: Scegli genitori basandosi sul fitness
4. **Crossover**: Combina genitori per creare figli
5. **Mutazione**: Applica cambiamenti casuali ai figli
6. **Sostituzione**: Forma nuova popolazione
7. Ripeti da passo 2 fino a convergenza

10.3 Rappresentazione e Operatori

Rappresentazione binaria: Il cromosoma più semplice è una stringa di bit. Per ottimizzare una funzione su $\mathbb{R}^n$, ogni variabile è codificata in binario.

Per esempio, per rappresentare un numero reale $x \in [a, b]$ con precisione di m bit:

- Dividi $[a, b]$ in 2^m intervalli
- Un intero $k \in [0, 2^m - 1]$ corrisponde a $x = a + k \cdot (b-a) / (2^m - 1)$
- k è rappresentato come stringa binaria di m bit

Selezione: Diversi metodi per scegliere i genitori:

Roulette wheel selection: Probabilità di selezione proporzionale al fitness: $p_i = \text{fitness}_i / \sum \text{fitness}_j$

Tournament selection: Scegli k individui casualmente, seleziona il migliore. Ripeti per ogni genitore necessario.

Rank selection: Ordina per fitness, probabilità basata sul rank invece del valore assoluto del fitness.

Crossover: Combina due genitori per creare figli:

Single-point crossover:

Genitore 1: 11001|01101

Genitore 2: 00110|11001

↓

Figlio 1: 11001|11001

Figlio 2: 00110|01101

Two-point crossover: Due punti di taglio invece di uno.

Uniform crossover: Ogni bit del figlio viene da genitore 1 o 2 con probabilità 0,5.

Mutazione: Flip casuale di bit con piccola probabilità p_m (tipicamente 0,001 - 0,01):

Prima: 110010110

Dopo: 110110110 (bit 3 flipato)

10.4 Schema Theorem e Analisi Teorica

Il comportamento degli algoritmi genetici può essere analizzato formalmente attraverso lo **Schema Theorem** di Holland. Uno **schema** è un pattern nel cromosoma dove alcuni bit sono specificati e altri sono "wildcard" (indicati con *).

Per esempio, con cromosomi di lunghezza 8:

- Schema: **10*1** rappresenta tutte le stringhe che hanno 1 in posizione 0, 0 in posizione 3, e 1 in posizione 5
- 10001100, 11001101, 10101111 appartengono tutti a questo schema

Definiamo:

- **Ordine** $o(H)$: numero di bit specificati (non wildcard)
- **Defining length** $\delta(H)$: distanza tra primo e ultimo bit specificato

Il teorema degli schemi afferma che schemi con alta fitness, basso ordine e bassa defining length ricevono un numero esponenzialmente crescente di copie in generazioni successive.

Matematicamente, se $m(H,t)$ è il numero di istanze dello schema H al tempo t , e $f(H)$ è il fitness medio dello schema:

$$m(H, t+1) \geq m(H,t) \cdot [f(H)/\bar{f}] \cdot [1 - p_c \cdot \delta(H)/(L-1)] \cdot [1 - p_m]^{o(H)}$$

dove $\bar{f}$ è il fitness medio della popolazione, p_c è la probabilità di crossover, p_m è la probabilità di mutazione, e L è la lunghezza del cromosoma.

Questo spiega perché i GA funzionano: costruiscono implicitamente "building blocks" (schemi) di buone soluzioni parziali e li combinano per creare soluzioni complete migliori.

10.5 Rappresentazioni Alternative

Rappresentazione con valori reali: Per ottimizzazione continua, usare direttamente vettori di numeri reali invece di stringhe binarie.

Operatori modificati:

- *Crossover aritmetico*: $child = \alpha \cdot parent1 + (1-\alpha) \cdot parent2$
- *Crossover BLX- α* : Crea figli in un intervallo esteso attorno ai genitori
- *Mutazione gaussiana*: $x' = x + N(0, \sigma)$

Rappresentazione a permutazione: Per problemi come TSP, il cromosoma è una permutazione di città.

Operatori speciali:

- *Order crossover (OX)*: Preserva l'ordine relativo
- *Partially mapped crossover (PMX)*: Scambia segmenti mantenendo validità
- *Swap mutation*: Scambia due elementi casuali

Rappresentazione ad albero: Per programmazione genetica, dove si evolve codice/espressioni.

10.6 Algoritmi Genetici Multi-Obiettivo

Molti problemi reali richiedono ottimizzazione di obiettivi multipli, spesso in conflitto. Gli algoritmi genetici si prestano naturalmente a questo attraverso il concetto di dominanza di Pareto.

Una soluzione x domina y ($x \succ y$) se:

- x è migliore o uguale a y in tutti gli obiettivi
- x è strettamente migliore di y in almeno un obiettivo

Il **NSGA-II** (Non-dominated Sorting Genetic Algorithm II) è l'algoritmo multi-obiettivo più popolare:

1. Classifica la popolazione in fronti di non-dominanza
2. Assegna fitness basato sul fronte (fronte 1 ha fitness più alto)
3. All'interno di ogni fronte, favorisce soluzioni più "sparse" (crowding distance)
4. Usa selezione basata su fronte e crowding distance

Questo mantiene diversità lungo il fronte di Pareto, fornendo all'utente un set di soluzioni di compromesso tra gli obiettivi.

10.7 Applicazioni degli Algoritmi Genetici

Progettazione ingegneristica: Ottimizzare forme di ali di aeroplani, strutture di ponti, circuiti integrati.

Scheduling: Assegnare task a risorse rispettando vincoli complessi (calendario scolastico, turni di lavoro, produzione manifatturiera).

Machine learning:

- Feature selection: scegliere subset ottimali di feature
- Architettura di reti neurali: evolvere topologie e iperparametri
- Ensemble methods: evolvere combinazioni di classificatori

Bioinformatica: Allineamento di sequenze, predizione di strutture proteiche, disegno di farmaci.

Finanza: Portfolio optimization, trading strategies, risk management.

Gaming e AI creativa: Evolvere comportamenti di NPC, generare contenuti procedurali, comporre musica.

La forza dei GA sta nella loro versatilità: possono gestire spazi di ricerca discreti, continui o misti, funzioni obiettivo non differenziabili, vincoli complessi, e obiettivi multipli. La parallela esplorazione di molteplici soluzioni li rende robusti e meno suscettibili a minimi locali rispetto a metodi di ricerca locale.

PUNTO 1: ASPETTI COMPUTAZIONALI MODERNI

Capitolo Aggiuntivo: Algoritmi Evolutivi Avanzati e Metodi Ibridi

10.8 Differential Evolution: Evoluzione attraverso Differenze Vettoriali

La Differential Evolution (DE), proposta da Rainer Storn e Kenneth Price nel 1995, rappresenta un'evoluzione concettuale rispetto agli algoritmi genetici tradizionali. Mentre i GA operano attraverso crossover bit a bit e mutazioni discrete, la DE sfrutta la geometria dello spazio di ricerca continuo attraverso operazioni vettoriali.

Il principio fondamentale

L'idea centrale è sorprendentemente semplice ed elegante: per esplorare lo spazio di ricerca, invece di applicare mutazioni casuali indipendenti, la DE utilizza le differenze tra individui esistenti nella popolazione come vettori di esplorazione. Questo approccio incorpora automaticamente informazione sulla distribuzione della popolazione nello spazio di ricerca.

Consideriamo una popolazione di N vettori x_i appartenenti a R^D (dove D è la dimensionalità del problema). Ad ogni generazione, per ciascun vettore target x_i , la DE genera un vettore mutante attraverso la strategia di mutazione.

Strategia DE/rand/1

La strategia più semplice e comunemente usata è DE/rand/1:

$$v_i = x_{r1} + F \times (x_{r2} - x_{r3})$$

dove $r1, r2, r3$ sono indici casuali distinti tra loro e diversi da i , e F è il fattore di scala della mutazione, tipicamente F appartenente a $[0.4, 1.0]$.

Questa formula ha una geometria intuitiva: prendiamo un vettore casuale x_{r1} come base, e lo perturbiamo nella direzione della differenza tra altri due vettori casuali. Il vettore differenza $(x_{r2} - x_{r3})$ codifica informazione sulla distribuzione della popolazione: se la popolazione è

concentrata in una regione, le differenze saranno piccole; se è dispersa, le differenze saranno grandi. Questo fornisce un meccanismo di auto-adattamento implicito.

Crossover binomiale

Dopo la mutazione, si applica il crossover tra il vettore target x_i e il vettore mutante v_i per generare il vettore trial u_i :

$$u_{i,j} = v_{i,j} \text{ se } \text{rand}_j \leq CR \text{ oppure } j = j_{\text{rand}} \text{ altrimenti } u_{i,j} = x_{i,j}$$

dove CR appartiene a $[0, 1]$ è il tasso di crossover e j_{rand} è un indice casuale che garantisce che almeno una componente provenga dal vettore mutante.

Selezione

La selezione nella DE è deterministica e basata sul confronto diretto:

$$x_i(t+1) = u_i \text{ se } f(u_i) \leq f(x_i) \text{ altrimenti } x_i(t+1) = x_i$$

Questa selezione golosa (greedy) garantisce che la popolazione non peggiori mai, una proprietà chiamata elitismo implicito.

Varianti strategiche

Esistono numerose varianti della strategia di mutazione, denotate con la nomenclatura DE/base/num_diff:

DE/best/1: $v_i = x_{\text{best}} + F \times (x_{r1} - x_{r2})$ Usa il miglior individuo come base, convergenza più rapida ma rischio di stagnazione.

DE/current-to-best/1: $v_i = x_i + F \times (x_{\text{best}} - x_i) + F \times (x_{r1} - x_{r2})$ Bilancia esplorazione e sfruttamento guidando verso il migliore.

DE/rand/2: $v_i = x_{r1} + F \times (x_{r2} - x_{r3}) + F \times (x_{r4} - x_{r5})$ Usa due vettori differenza, maggiore diversità esplorativa.

Analisi della convergenza

La convergenza della DE può essere analizzata attraverso la teoria delle catene di Markov. Definiamo lo stato del sistema come la popolazione intera $P(t) = \{x_1(t), \dots, x_N(t)\}$. L'evoluzione è stocastica a causa delle selezioni casuali di $r1, r2, r3$.

Un risultato importante, dimostrato da Zaharie, è che la varianza della popolazione $\sigma^2(t)$ evolve approssimativamente secondo:

$$\sigma^2(t+1) \approx \sigma^2(t) \times [1 - 2F^2 \times CR \times (1 - CR)]$$

Per convergenza, vogliamo $\sigma^2(t) \rightarrow 0$, il che richiede:

$$F^2 \times CR \times (1 - CR) < 0.5$$

Questo fornisce una condizione sufficiente per la convergenza (anche se non necessaria, poiché l'analisi è approssimata).

Bilancio esplorazione-sfruttamento

Il parametro F controlla l'ampiezza delle perturbazioni: F alto favorisce l'esplorazione (passi grandi), F basso favorisce lo sfruttamento (raffinamento locale). Il parametro CR controlla quante componenti vengono ereditate dal vettore mutante: CR alto implica maggiore esplorazione, CR basso maggiore conservazione.

Un'analisi più sofisticata mostra che il rapporto ottimale tra esplorazione e sfruttamento varia durante l'ottimizzazione: all'inizio vogliamo esplorare ampiamente, verso la fine raffinare localmente. Questo ha portato allo sviluppo di strategie adattative.

Self-Adaptive Differential Evolution (SaDE)

In SaDE, i parametri F e CR non sono costanti ma evolvono insieme alla popolazione. Ogni individuo x_i ha associato un proprio F_i e CR_i . Questi parametri sono perturbati e soggetti a selezione insieme ai vettori soluzione.

Una strategia comune è:

$$F_i = F_{\min} + \text{rand}_i \times (F_{\max} - F_{\min}) \quad CR_i = \text{randn}_i(\mu_{CR}, \sigma_{CR})$$

dove μ_{CR} e σ_{CR} sono aggiornati in base al successo delle generazioni precedenti usando una media mobile.

JADE (Adaptive Differential Evolution with Optional External Archive)

JADE, proposto da Zhang e Sanderson, introduce meccanismi sofisticati di adattamento:

1. I parametri F e CR sono generati da distribuzioni di Cauchy e normale rispettivamente, con parametri centrali aggiornati in base ai parametri che hanno prodotto successo
2. Un archivio esterno mantiene soluzioni recentemente sostituite, che possono essere usate nella mutazione per aumentare la diversità
3. La strategia di mutazione è DE/current-to-pbest, che usa non solo il migliore assoluto ma uno dei $p\%$ migliori, evitando convergenza prematura

L'aggiornamento dei parametri è:

$$\mu_{CR}(t+1) = (1 - c) \times \mu_{CR}(t) + c \times \text{mean}_A(CR_{\text{successo}}) \quad \mu_F(t+1) = (1 - c) \times \mu_F(t) + c \times \text{mean}_L(F_{\text{successo}})$$

dove c è il tasso di apprendimento, mean_A è la media aritmetica e mean_L è la media di Lehmer definita come:

$$\text{mean_L}(F) = \Sigma(F_k^2) / \Sigma(F_k)$$

che pesa maggiormente i valori più grandi.

10.9 Firefly Algorithm: Ottimizzazione attraverso Attrattività Luminosa

L'algoritmo Firefly, sviluppato da Xin-She Yang nel 2008, si ispira al comportamento di bioluminescenza delle lucciole. Il principio è che le lucciole sono attratte da quelle più luminose, dove la luminosità rappresenta la qualità della soluzione.

Formulazione matematica

Ogni lucciola i alla posizione x_i ha una luminosità $I(x_i)$ proporzionale al valore della funzione obiettivo. La luminosità percepita dalla lucciola j dalla lucciola i è:

$$I(r) = I_0 / (1 + \gamma r^2)$$

dove $r = \|x_i - x_j\|$ è la distanza euclidea e γ è il coefficiente di assorbimento della luce nel medium (che modella come la luce si attenua con la distanza).

L'attrattività β diminuisce con la distanza:

$$\beta(r) = \beta_0 \times \exp(-\gamma r^2)$$

dove β_0 è l'attrattività a distanza zero.

Equazione di movimento

Se la lucciola i è meno luminosa della lucciola j , si muove verso j secondo:

$$x_i(t+1) = x_i(t) + \beta_0 \times \exp(-\gamma r_{ij}^2) \times (x_j(t) - x_i(t)) + \alpha \times \varepsilon_i$$

dove:

- Il primo termine è la posizione attuale
- Il secondo termine rappresenta l'attrazione verso la lucciola più luminosa
- Il terzo termine è un componente di randomizzazione con α parametro di controllo e ε_i un vettore casuale da una distribuzione (tipicamente uniforme o gaussiana)

Proprietà emergenti

L'algoritmo ha proprietà interessanti:

1. **Suddivisione automatica:** Lucciole si raggruppano automaticamente attorno a ottimi locali multipli, formando sotto-sciami. Questo è utile per problemi multimodali.
2. **Regolazione automatica della scala:** Il parametro γ controlla la topologia di interazione. Con γ grande, l'algoritmo si comporta come ricerca locale parallela. Con γ piccolo, si comporta come PSO.

3. **Convergenza garantita:** Per $\alpha \rightarrow 0$ nel tempo e con parametri appropriati, la convergenza al miglior individuo è garantita.

Analisi di convergenza

Possiamo analizzare il comportamento asintotico. Nel limite $\gamma \rightarrow 0$ (nessun assorbimento), ogni lucciola è attratta dalla migliore globale, e l'algoritmo diventa:

$$x_i(t+1) \approx x_i(t) + \beta_0(x_{\text{best}} - x_i(t)) + \alpha \times \varepsilon_i$$

Questo è equivalente a una dinamica di tipo PSO. Prendendo il valore atteso:

$$E[x_i(t+1)] = (1 - \beta_0) \times x_i(t) + \beta_0 \times x_{\text{best}}$$

che converge esponenzialmente a x_{best} con tasso $(1 - \beta_0)$.

Nel limite $\gamma \rightarrow \infty$ (forte assorbimento), ogni lucciola interagisce solo con lucciole molto vicine, risultando in ricerca locale parallela indipendente.

10.10 Bat Algorithm: Ecolocalizzazione e Ottimizzazione

L'algoritmo Bat, anch'esso di Xin-She Yang (2010), si ispira al comportamento di ecolocalizzazione dei pipistrelli che cercano prede.

Principi biologici

I pipistrelli emettono impulsi ultrasonici e ascoltano gli echi per localizzare prede e ostacoli. Regolano la frequenza, il volume e il tasso di emissione degli impulsi in base alla vicinanza alla preda.

Modellazione matematica

Ogni pipistrello i ha posizione x_i , velocità v_i , e emette impulsi con frequenza f_i , volume A_i e tasso di emissione r_i .

Ad ogni iterazione:

1. Genera frequenza casuale: $f_i = f_{\text{min}} + (f_{\text{max}} - f_{\text{min}}) \times \beta$ dove β appartiene a $[0,1]$ è casuale
2. Aggiorna velocità e posizione: $v_i(t) = v_i(t-1) + (x_i(t-1) - x_{\text{best}}) \times f_i$
 $x_i(t) = x_i(t-1) + v_i(t)$
3. Se $\text{rand} > r_i$, genera soluzione locale attorno alla migliore corrente: $x_{\text{new}} = x_{\text{best}} + \varepsilon \times A_{\text{media}}(t)$ dove ε appartiene a $[-1, 1]$ e A_{media} è il volume medio corrente
4. Se $\text{rand} < A_i$ e $f(x_{\text{new}}) < f(x_i)$:
 - Accetta la nuova soluzione

- Aumenta r_i : $r_i(t+1) = r_i(0) \times [1 - \exp(-\gamma t)]$
- Diminuisce A_i : $A_i(t+1) = \alpha \times A_i(t)$

dove α e γ sono costanti (tipicamente $\alpha = \gamma = 0.9$).

Interpretazione

Il volume A_i rappresenta l'intensità dell'esplorazione (volumi alti = esplorazione ampia). Il tasso di emissione r_i rappresenta la probabilità di ricerca locale (r_i alto = più ricerca locale).

All'inizio: A_i alto, r_i basso $\rightarrow$ esplorazione globale Verso la fine: A_i basso, r_i alto $\rightarrow$ raffinamento locale

Questo scheduling automatico del bilancio esplorazione-sfruttamento è simile al simulated annealing.

Confronto teorico

Yang ha dimostrato che nel limite di parametri appropriati:

- Con $r_i \rightarrow 0$ per tutti, BA si comporta come PSO
- Con $A_i \rightarrow 0$ per tutti, BA diventa una ricerca locale stocastica
- Il comportamento generale è una transizione smooth tra questi estremi

10.11 Algoritmi Ibridi: Sinergia tra Metodi Esatti e Metaeuristiche

L'osservazione chiave è che i metodi esatti (come branch-and-bound, programmazione dinamica) e le metaeuristiche (GA, PSO, ACO, ecc.) hanno punti di forza complementari:

Metodi esatti:

- Pro: Garantiscono ottimalità (se completano), forniscono bounds di qualità
- Contro: Tempo esponenziale nel caso peggiore, non scalano su istanze grandi

Metaeuristiche:

- Pro: Trovano rapidamente buone soluzioni, scalano bene
- Contro: Nessuna garanzia di ottimalità, difficile valutare la qualità

Gli algoritmi ibridi cercano di combinare il meglio di entrambi i mondi.

Categorie di ibridazione

1. Metaeuristiche per generare soluzioni iniziali per metodi esatti

Esempio: Genetic Algorithm + Branch-and-Bound per il TSP

Procedura:

1. Esegui GA per N generazioni $\rightarrow$ ottieni soluzione incumbent x_{GA} con costo C_{GA}

2. Usa x_{GA} e C_{GA} come punto di partenza per branch-and-bound
3. Il B&B può potare rami con $\text{bound} \geq C_{GA}$ più aggressivamente

Teorema: Se il GA trova una soluzione entro $k\%$ dell'ottimo, il B&B esplora al massimo V_k nodi, dove V_k può essere ordini di grandezza inferiore all'albero completo.

2. Metodi esatti per risolvere sotto-problemi in metaeuristiche

Esempio: Memetic Algorithm = GA + Ricerca Locale Ottimale

Idea: Dopo crossover e mutazione, non valutare immediatamente il fitness, ma prima risolvi esattamente un sotto-problema locale.

Per il TSP:

1. GA genera tour T
2. Per ogni città i , risolvi esattamente il problema: qual è il migliore 2-opt o 3-opt centrato su i ?
3. Applica la migliore mossa locale trovata
4. Valuta il fitness del tour ottimizzato localmente

Questo è costoso ma riduce drasticamente il numero di generazioni necessarie.

3. Decomposizione del problema

Large Neighborhood Search (LNS) ibrido:

Idea: Dividi il problema in parti sovrapposte, risolvi ciascuna parte esattamente, ricombina.

Per il Vehicle Routing Problem:

1. PSO genera un routing completo
2. Identifica sotto-regioni di k clienti
3. Per ogni sotto-regione, risolvi esattamente il sotto-problema VRP usando programmazione intera mista
4. Ricombina le soluzioni ottimali locali
5. Usa il risultato come particella nel PSO

4. Matheuristics: integrazione stretta con programmazione matematica

Fix-and-Optimize per problemi di programmazione intera mista (MIP):

Data MIP: $\min c^T x$ soggetto a $Ax \leq b$, x_i intero per i in I

Procedura ibrida GA-MIP:

1. GA mantiene popolazione di vettori che partizionano l'insieme I in $I_1, I_2, \dots, I_k$
2. Per ogni partizione P nella popolazione: a. Fissa x_j per j non in I_1 ai valori della soluzione incumbent b. Risolvi il MIP ridotto dove solo x_i per i in I_1 sono variabili c. Aggiorna incumbent se miglioramento d. Ripeti per $I_2, \dots, I_k$ ciclicamente
3. GA evolve le partizioni usando crossover e mutazione

5. Parallel Cooperative Search

Esegui in parallelo:

- Thread 1: Branch-and-Bound esatto
- Thread 2: Genetic Algorithm
- Thread 3: Simulated Annealing
- Thread 4: Tabu Search

Con comunicazione:

- Quando GA trova nuova migliore soluzione → invia a B&B come nuovo bound superiore
- Quando B&B trova bound inferiore migliorato → invia a metaeuristiche come informazione
- Metaeuristiche scambiano periodicamente le migliori soluzioni trovate

Questo approccio ha dato risultati eccellenti in competizioni di ottimizzazione.

Analisi teorica dell'ibridazione

Teorema (approssimato): Se una metaeuristica M ha probabilità p_M di trovare una soluzione entro ε dell'ottimo in tempo T_M , e un metodo esatto E risolve il problema esattamente in tempo $T_E = T_E(\text{gap_iniziale})$, allora l'ibrido $M+E$ ha tempo atteso:

$$T_{\text{ibrido}} \approx T_M + E[T_E \mid \text{gap} = \varepsilon] < T_E(\text{gap} = \infty)$$

dove $E[T_E \mid \text{gap} = \varepsilon]$ è il tempo atteso di E quando inizia con gap ε .

Per molti problemi, $T_E(\text{gap})$ decresce esponenzialmente con la riduzione del gap:

$$T_E(\text{gap}) \propto \exp(-\lambda \times (\text{gap_max} - \text{gap}))$$

quindi anche piccole riduzioni del gap iniziale possono dare enormi speed-up.

Esempio concreto: TSP Ibrido

Consideriamo il Travelling Salesman Problem su $n = 100$ città.

Algoritmo ibrido GA-LKH:

1. GA con popolazione 100, 500 generazioni
2. Ogni 50 generazioni, applica Lin-Kernighan-Helsgaun (LKH) al miglior individuo
3. LKH è un metodo di ricerca locale sofisticato che risolve "quasi ottimamente" il TSP

Risultati tipici:

- GA puro: trova soluzioni entro 5% dell'ottimo in 500 generazioni
- LKH puro: trova ottimo o quasi-ottimo ma richiede inizializzazione buona
- GA+LKH: trova ottimo o entro 0.1% in 200 generazioni equivalenti

Il costo computazionale dell'ibridazione è ripagato dalla convergenza più rapida e dalla qualità superiore.

Direzioni di ricerca

Gli algoritmi ibridi rappresentano la frontiera pratica dell'ottimizzazione combinatoria. Direzioni attive:

- Apprendimento automatico per decidere quando applicare quale componente
- Auto-tuning dei parametri di ibridazione
- Parallelizzazione su GPU di componenti ibride
- Ibridazione con deep learning per problemi di ottimizzazione su grafi

PARTE IV: APPLICAZIONI MODERNE

Capitolo 11: Ottimizzazione di Percorsi e Logistica

11.1 GPS e Routing in Tempo Reale

I moderni sistemi di navigazione GPS affrontano un problema di ottimizzazione dinamica: trovare il percorso ottimale tra un'origine e una destinazione, considerando condizioni di traffico in tempo reale, lavori stradali, incidenti e preferenze dell'utente.

Il problema è formulato come ricerca del percorso di costo minimo in un grafo pesato. La rete stradale è rappresentata come un grafo $G = (V, E)$ dove:

- V è l'insieme dei nodi (intersezioni)
- E è l'insieme degli archi (strade)
- Ogni arco (i, j) ha un peso $w_{\{ij\}}$ che rappresenta il "costo" di percorrerlo

Il costo può essere definito in diversi modi:

- **Distanza fisica:** $w_{\{ij\}}$ = lunghezza della strada
- **Tempo di percorrenza:** $w_{\{ij\}}$ = lunghezza / velocità media
- **Costo generalizzato:** combinazione di tempo, pedaggio, consumo carburante, preferenze

11.2 L'Algoritmo di Dijkstra

L'algoritmo di Dijkstra (1959) è il metodo classico per trovare il percorso più breve da una sorgente a tutti gli altri nodi in un grafo con pesi non negativi.

Principio: Costruisce progressivamente un albero di percorsi più brevi, espandendo sempre il nodo più vicino alla sorgente che non è ancora stato processato.

Pseudocodice:

Dijkstra(G, s):

$\text{dist}[s] = 0$

 per tutti i nodi $v \neq s$:

$\text{dist}[v] = \infty$

$Q =$ insieme di tutti i nodi

 mentre Q non è vuoto:

$u =$ nodo in Q con $\text{dist}[u]$ minima

 rimuovi u da Q

 per ogni vicino v di u :

$\text{alt} = \text{dist}[u] + \text{peso}(u, v)$

 se $\text{alt} < \text{dist}[v]$:

$\text{dist}[v] = \text{alt}$

$\text{predecessore}[v] = u$

Complessità: Con una heap binaria per la coda di priorità, $O((|V| + |E|) \log |V|)$. Con una heap di Fibonacci, $O(|E| + |V| \log |V|)$.

11.3 A* e Ricerca Euristica

L'algoritmo A* (pronunciato "A-star"), sviluppato da Hart, Nilsson e Raphael nel 1968, migliora Dijkstra utilizzando informazione euristica sulla posizione del goal.

A* mantiene per ogni nodo n :

- $g(n)$: costo del percorso migliore trovato finora da start a n
- $h(n)$: stima euristica del costo da n a goal
- $f(n) = g(n) + h(n)$: stima del costo totale del percorso attraverso n

Proprietà dell'euristica:

- **Ammissibile:** $h(n) \leq$ costo reale da n a goal
- **Consistente:** $h(n) \leq \text{costo}(n, n') + h(n')$ per ogni vicino n'

Se h è ammissibile, A* trova sempre il percorso ottimale.

Euristica comune per routing stradale: $h(n) = \text{distanza_euclidea}(n, \text{goal}) / \text{velocità_massima}$

Questa è ammissibile perché sottostima sempre il tempo reale necessario.

11.4 Ottimizzazione di Flotte di Veicoli

Il **Vehicle Routing Problem (VRP)** generalizza il TSP: una flotta di veicoli deve servire clienti distribuiti geograficamente, partendo da uno o più depositi, rispettando vincoli di capacità e finestre temporali.

Formulazione matematica: Dato:

- Insieme di clienti $C = \{1, 2, \dots, n\}$
- Deposito(i) D
- Flotta di m veicoli, ciascuno con capacità Q
- Domanda d_i di ogni cliente i
- Costo c_{ij} per viaggiare da i a j

Variabili decisionali:

- $x_{ijk} = 1$ se il veicolo k viaggia direttamente da i a j, 0 altrimenti

Obiettivo: Minimizzare il costo totale: $\min \sum_{i,j,k} c_{ij} x_{ijk}$

Vincoli:

1. Ogni cliente è visitato esattamente una volta: $\sum_{j,k} x_{ijk} = 1$ per ogni $i \in C$
2. Conservazione del flusso (un veicolo che arriva deve partire): $\sum_j x_{ijk} = \sum_j x_{jik}$ per ogni i, k
3. Capacità dei veicoli non superata: $\sum_i d_i (\sum_j x_{ijk}) \leq Q$ per ogni k
4. Ogni route inizia e finisce al deposito

Questo è un problema NP-hard, ancora più difficile del TSP. Metodi esatti funzionano solo per istanze piccole (< 100 clienti). Per istanze reali, si usano euristiche e metaeuristiche.

11.5 Algoritmi per VRP

Clarke-Wright Savings Algorithm (1964): Euristica costruttiva basata sul concetto di "risparmio".

Il risparmio s_{ij} di combinare due route (deposito $\rightarrow i \rightarrow$ deposito) e (deposito $\rightarrow j \rightarrow$ deposito) in una singola route (deposito $\rightarrow i \rightarrow j \rightarrow$ deposito) è:

$$s_{ij} = c_{0i} + c_{0j} - c_{ij}$$

dove 0 indica il deposito.

Algoritmo:

1. Inizia con n route separate, una per cliente
2. Calcola s_{ij} per tutte le coppie
3. Ordina i risparmi in ordine decrescente
4. Per ogni risparmio s_{ij} (in ordine):
 - Se i e j sono in route diverse e possono essere uniti rispettando vincoli di capacità:
 - Unisci le due route

Metaeuristiche:

- **Tabu Search:** Ricerca locale con memoria per evitare cicli
- **Simulated Annealing:** Accetta occasionalmente soluzioni peggiori per sfuggire a ottimi locali
- **Genetic Algorithms:** Evolve popolazione di soluzioni
- **Ant Colony Optimization:** Come visto prima, adattato per VRP

Algoritmi ibridi: Combinano metodi esatti per subproblemi con euristiche per il problema globale.

11.6 Routing Dinamico con Informazioni in Tempo Reale

I sistemi moderni di routing devono adattarsi a condizioni mutevoli:

- **Congestione del traffico:** I pesi degli archi cambiano in tempo reale
- **Richieste dinamiche:** Nuovi clienti/destinazioni appaiono durante l'esecuzione
- **Eventi imprevisti:** Incidenti, lavori stradali, maltempo

Approccio di re-ottimizzazione:

1. Calcola route iniziali
2. Monitora condizioni in tempo reale
3. Quando avviene un cambiamento significativo:
 - Re-calcola le route da posizioni correnti dei veicoli
 - Comunica aggiornamenti ai conducenti

Ottimizzazione predittiva: Utilizza modelli storico per prevedere condizioni future (traffico a certe ore, domanda stagionale) e ottimizzare preventivamente.

Reinforcement Learning: Algoritmi come Deep Q-Networks (DQN) possono imparare politiche di routing da esperienze passate, adattandosi a pattern complessi che modelli analitici non catturano facilmente.

Capitolo 12: Protein Folding e Biologia Computazionale**12.1 Il Problema del Ripiegamento Proteico**

Le proteine sono catene di amminoacidi che si ripiegano in strutture tridimensionali specifiche. La funzione di una proteina dipende criticamente dalla sua struttura 3D. Predire questa struttura dalla sequenza amminoacidica è uno dei problemi più importanti in biologia.

Paradosso di Levinthal: Una proteina tipica ha centinaia di amminoacidi. Se ciascun amminoacido può assumere solo 3 configurazioni (una semplificazione drastica), una proteina di 100 amminoacidi ha $3^{100} \approx 5 \times 10^{47}$ configurazioni possibili. Anche esplorando un miliardo di configurazioni al secondo, ci vorrebbero più dell'età dell'universo per provarle tutte. Eppure le proteine si ripiegano spontaneamente in millisecondi.

La soluzione al paradosso è che il ripiegamento non è una ricerca casuale, ma è guidato dal principio di minima energia libera. La struttura nativa è quella che minimizza l'energia libera di Gibbs.

12.2 Modelli di Energia

L'energia di una configurazione proteica include diversi contributi:

Interazioni di van der Waals: $E_{\text{vdW}} = \sum_{\{i < j\}} \epsilon_{\{ij\}} \left[\left(\frac{r_{\text{min},ij}}{r_{\{ij\}}} \right)^{12} - 2 \left(\frac{r_{\text{min},ij}}{r_{\{ij\}}} \right)^6 \right]$

dove $r_{\{ij\}}$ è la distanza tra atomi i e j .

Interazioni elettrostatiche: $E_{\text{elec}} = \sum_{\{i < j\}} (q_i q_j) / (4\pi\epsilon_0 r_{\{ij\}})$

dove q_i è la carica parziale dell'atomo i .

Legami idrogeno: Contributo energetico favorevole quando geometria appropriata tra donatore e accettore.

Interazioni idrofobiche: Amminoacidi idrofobici tendono a raggrupparsi all'interno della proteina, evitando l'acqua.

Energia conformazionale: Energia associata agli angoli di torsione (ϕ , ψ) della catena principale.

Energia totale: $E_{\text{tot}} = E_{\text{vdW}} + E_{\text{elec}} + E_{\text{Hbond}} + E_{\text{hydrophobic}} + E_{\text{torsion}} + E_{\text{altri}}$

Il problema è minimizzare E_{tot} rispetto alle coordinate di tutti gli atomi.

12.3 Algoritmi di Minimizzazione dell'Energia

Minimizzazione locale: Gradient descent, conjugate gradient, L-BFGS. Partendo da una configurazione iniziale, si scende lungo il gradiente negativo dell'energia. Veloce ma trova solo ottimi locali.

Dinamica Molecolare (MD): Simula la dinamica fisica della proteina integrando le equazioni di Newton:

$$m_i \frac{d^2 r_i}{dt^2} = -\nabla_i E$$

Permette di esplorare lo spazio conformazionale attraverso barriere energetiche, ma è computazionalmente intensivo.

Monte Carlo: Genera configurazioni casuali e le accetta/rifiuta basandosi sull'energia, secondo il criterio di Metropolis:

$$P(\text{accetta}) = \min(1, e^{\{-\Delta E / (k_B T)\}})$$

dove ΔE è il cambiamento di energia, k_B è la costante di Boltzmann, e T è la temperatura.

Simulated Annealing: Versione di Monte Carlo con temperatura decrescente. All'inizio, alta temperatura permette ampia esplorazione. Gradualmente si raffredda, convergendo verso minimo globale.

Algoritmi genetici: Popolazione di conformazioni evolve attraverso crossover e mutazione, con fitness = $-E$.

12.4 AlphaFold: Deep Learning per Protein Structure

Nel 2020, DeepMind ha rivoluzionato il campo con AlphaFold2, un sistema di deep learning che predice strutture proteiche con accuratezza quasi sperimentale.

Architettura:

1. **Embedding iniziale:** Sequenza amminoacidica + MSA (multiple sequence alignment) → rappresentazioni ad alta dimensionalità
2. **Evoformer:** Transformer modificato che raffina:
 - Rappresentazioni della sequenza (embedding amminoacidi)
 - Rappresentazioni delle coppie (relazioni tra amminoacidi)
3. Attraverso attenzione multi-testa su righe, colonne e triangoli della matrice di coppie.
4. **Structure module:** Predice posizioni 3D degli atomi, iterativamente affinando attraverso "recycling" (output viene re-immesso come input).
5. **Loss function:** Combinazione di:
 - FAPE (Frame Aligned Point Error): allineamento strutturale locale
 - Distogram loss: predizione di distanze tra coppie
 - Masked MSA loss: predizione di residui mascherati

Innovazioni chiave:

- Attenzione su strutture geometriche (triangoli) oltre che sequenze
- Rappresentazione equivariante delle rotazioni
- Recycling iterativo
- Uso estensivo di MSA evolutive

AlphaFold non risolve direttamente l'ottimizzazione energetica, ma impara implicitamente il paesaggio energetico da milioni di strutture note, predicendo direttamente la struttura di minima energia.

12.5 Protein Design: Il Problema Inverso

Il protein design ribalta il problema: data una funzione desiderata (o una struttura target), quale sequenza amminoacidica la realizzerà?

Questo è un problema di ottimizzazione ancora più difficile:

- Spazio di ricerca: 20^n sequenze per proteina di lunghezza n
- Funzione obiettivo complessa: stabilità, solubilità, funzionalità

Approcci:

Metodi basati su energia: Minimizzare $E(\text{sequenza} \mid \text{struttura})$ dove la struttura è fissa e si ottimizza la sequenza.

Usa algoritmi come:

- Dead-end elimination
- Simulated annealing su sequenze
- MCMC con mosse di mutazione

RosettaDesign: Suite software che combina modellazione energetica con ricerca Monte Carlo. Ha prodotto proteine funzionali de novo verificate sperimentalmente.

Deep learning generativo: Modelli come ProteinMPNN usano reti neurali grafiche per "scrivere" sequenze compatibili con strutture date.

Approccio:

1. Encoder: struttura 3D $\rightarrow$ embedding
2. Decoder autoregressivo: genera sequenza amminoacido per amminoacido
3. Training: massimizza $P(\text{sequenza} \mid \text{struttura})$ su database di proteine note

Risultati impressionanti: sequenze progettate hanno successo sperimentale $> 50\%$, vs $< 10\%$ con metodi precedenti.

Capitolo 13: Machine Learning e Gradient Descent

13.1 L'Apprendimento come Ottimizzazione

Il machine learning riduce l'apprendimento a un problema di ottimizzazione: trovare i parametri θ di un modello che minimizzano una funzione di perdita (loss function) $L(\theta)$ sui dati di training.

Per una rete neurale con dati di training $\{(x_i, y_i)\}$:

$$L(\theta) = (1/N) \sum_i \ell(f(x_i; \theta), y_i) + \lambda R(\theta)$$

dove:

- $f(x; \theta)$ è l'output della rete con parametri θ
- ℓ è la loss per singolo esempio (es. MSE, cross-entropy)
- $R(\theta)$ è un termine di regolarizzazione (es. L2: $\|\theta\|^2$)
- λ controlla la forza della regolarizzazione

13.2 Gradient Descent e Varianti

Gradient Descent (GD): $\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t)$

dove η è il learning rate.

Calcola il gradiente sull'intero dataset. Accurato ma lento per grandi dataset.

Stochastic Gradient Descent (SGD): $\theta_{t+1} = \theta_t - \eta \nabla \ell(f(x_i; \theta_t), y_i)$

Aggiorna usando un singolo esempio alla volta. Rumoroso ma veloce, può sfuggire a minimi locali.

Mini-batch SGD: $\theta_{t+1} = \theta_t - \eta (1/B) \sum_{i \in \text{batch}} \nabla \ell(f(x_i; \theta_t), y_i)$

Compromesso: calcola gradiente su mini-batch di B esempi. Più stabile di SGD, più veloce di GD. Standard de facto.

13.3 Ottimizzatori Avanzati

Momentum: Accelera nella direzione di gradienti consistenti, smorza oscillazioni:

$$v_{t+1} = \beta v_t + \nabla L(\theta_t) \quad \theta_{t+1} = \theta_t - \eta v_{t+1}$$

Tipicamente $\beta = 0,9$.

Nesterov Accelerated Gradient (NAG): "Guarda avanti" prima di calcolare il gradiente:

$$v_{t+1} = \beta v_t + \nabla L(\theta_t - \eta \beta v_t) \quad \theta_{t+1} = \theta_t - \eta v_{t+1}$$

AdaGrad: Adatta il learning rate per ogni parametro basandosi sui gradienti storici:

$$g_{t,i} = g_{t-1,i} + (\nabla L(\theta_t))_i^2 \quad \theta_{t+1,i} = \theta_{t,i} - \eta / \sqrt{g_{t,i} + \epsilon} \times \nabla L(\theta_t)_i$$

Utile per dati sparsi, ma il learning rate può decadere troppo rapidamente.

RMSProp: Corregge il decadimento di AdaGrad usando media mobile esponenziale:

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma)(\nabla L(\theta_t))^2 \quad \theta_{t+1} = \theta_t - \eta / \sqrt{E[g^2]_t + \epsilon} \times \nabla L(\theta_t)$$

Tipicamente $\gamma = 0,9$.

Adam (Adaptive Moment Estimation): Combina momentum e RMSProp:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) \nabla L(\theta_t) \quad v_t = \beta_2 v_{t-1} + (1-\beta_2) (\nabla L(\theta_t))^2$$

$$\text{Correzione del bias: } \hat{m}_t = m_t / (1 - \beta_1^t) \quad \hat{v}_t = v_t / (1 - \beta_2^t)$$

$$\text{Aggiornamento: } \theta_{t+1} = \theta_t - \eta \hat{m}_t / \sqrt{\hat{v}_t + \epsilon}$$

Default: $\beta_1=0,9$, $\beta_2=0,999$, $\eta=0,001$. Ottimizzatore più popolare per deep learning.

13.4 Tecniche di Regularizzazione come Ottimizzazione Vincolata

La regularizzazione previene l'overfitting aggiungendo vincoli ai parametri del modello.

L2 Regularization (Weight Decay): Minimizza: $L(\theta) + \lambda ||\theta||^2$

Equivalente a vincolo: $||\theta||^2 \leq C$

Soluzione in forma chiusa per regressione lineare: $\theta = (X^T X + \lambda I)^{-1} X^T y$

Effetto: Riduce magnitudine dei pesi, favorisce soluzioni "smooth".

L1 Regularization (Lasso): Minimizza: $L(\theta) + \lambda ||\theta||_1$

Promuove sparsità: molti pesi diventano esattamente zero. Utile per feature selection.

Dropout: Durante training, "spegne" casualmente neuroni con probabilità p . Equivalente a ottimizzare un ensemble di reti più piccole. Previene co-adattamento eccessivo tra neuroni.

Early Stopping: Monitora performance su validation set, ferma training quando smette di migliorare. Forma implicita di regularizzazione: limita il numero di iterazioni di ottimizzazione.

13.5 Ottimizzazione di Iperparametri

Gli iperparametri (learning rate, architettura, regularizzazione, ecc.) non vengono appresi dal gradient descent ma devono essere ottimizzati separatamente.

Grid Search: Prova tutte le combinazioni in una griglia predefinita. Esaustivo ma costoso: $O(n^d)$ per n valori e d iperparametri.

Random Search: Campiona casualmente da distribuzioni di iperparametri. Sorprendentemente efficace: Bergstra & Bengio (2012) mostrarono che spesso supera grid search con stesso budget computazionale.

Bayesian Optimization: Modella $P(\text{performance} \mid \text{iperparametri})$ con un processo gaussiano. Usa acquisition function (es. Expected Improvement) per decidere quale configurazione provare successivamente. Efficiente: converge con poche valutazioni.

Hyperband: Combina random search con early stopping adattivo. Alloca più risorse (epoche) a configurazioni promettenti.

Neural Architecture Search (NAS): Ottimizza l'architettura stessa della rete. Usa reinforcement learning, evolutionary algorithms, o gradient-based methods. Computazionalmente intensivo ma ha prodotto architetture state-of-the-art.

Conclusione Generale

La Convergenza di Natura, Matematica e Computazione

Questo trattato ha esplorato un tema profondo e unificante: l'ottimizzazione come principio fondamentale che attraversa la natura, la matematica e la tecnologia. Abbiamo iniziato con i fondamenti teorici del calcolo variazionale, scoprendo come problemi apparentemente diversi - dalla brachistocrona alla catenaria, dalle superfici minime alle geodetiche - sono tutti unificati sotto il framework matematico dell'ottimizzazione funzionale.

Le equazioni di Eulero-Lagrange non sono mere astrazioni matematiche, ma descrivono leggi fisiche reali. Quando una bolla di sapone assume la sua forma sferica, sta risolvendo un problema variazionale. Quando la luce viaggia da un punto a un altro, minimizza il tempo di percorrenza secondo il principio di Fermat. Quando un sistema fisico evolve, segue il principio di minima azione. La natura "calcola" soluzioni ottimali non attraverso algoritmi espliciti, ma attraverso i processi fisici stessi che sono governati da questi principi.

Nella seconda parte, abbiamo visto come questi principi si manifestano concretamente nei sistemi biologici. L'esagono perfetto delle celle dell'alveare non è una coincidenza estetica ma la soluzione matematicamente ottimale al problema di massimizzare lo spazio con minimo materiale. La spirale aurea nella disposizione delle foglie e dei semi non è misticismo numerologico ma la configurazione che massimizza l'esposizione alla luce evitando ombreggiamento. La legge di Murray che governa i vasi sanguigni emerge dal bilanciamento tra costo metabolico e efficienza del trasporto.

Questi esempi rivelano una verità profonda: l'evoluzione è essa stessa un processo di ottimizzazione. Attraverso variazione, selezione ed eredità, gli organismi convergono verso soluzioni che, se non sempre perfettamente ottimali in senso matematico assoluto, rappresentano compromessi eccellenti tra vincoli multipli e spesso in conflitto. La pressione selettiva favorisce configurazioni che risolvono efficientemente problemi di ottimizzazione multi-obiettivo: sopravvivenza, riproduzione, efficienza energetica, robustezza.

Nella terza parte, abbiamo visto come gli scienziati informatici hanno tratto ispirazione da questi processi naturali per creare algoritmi potenti e versatili. Gli algoritmi di Ant Colony Optimization imitano il comportamento collettivo delle formiche per risolvere problemi combinatori complessi come il TSP. Il Particle Swarm Optimization cattura l'intelligenza collettiva di stormi e banchi per esplorare efficientemente spazi di ricerca continui. Gli algoritmi genetici applicano i principi darwiniani di evoluzione alla ricerca di soluzioni ottimali.

Questi algoritmi bio-ispirati non sono semplici curiosità accademiche. Sono strumenti pratici ampiamente utilizzati in ingegneria, logistica, finanza, progettazione e molti altri campi. La loro forza risiede nella capacità di gestire problemi dove metodi tradizionali falliscono: spazi di ricerca vasti e complessi, funzioni obiettivo non differenziabili o rumorose, vincoli multipli, obiettivi in conflitto.

Nell'ultima parte, abbiamo esplorato applicazioni moderne dove l'ottimizzazione è centrale. I sistemi GPS che usiamo quotidianamente risolvono in tempo reale problemi di shortest path su grafi con milioni di nodi. AlphaFold ha rivoluzionato la biologia strutturale usando deep learning per predire strutture proteiche che minimizzano energia libera. Il machine learning stesso è fondamentalmente un problema di ottimizzazione: trovare parametri che minimizzano errore di predizione.

Il gradient descent e le sue varianti sofisticate (momentum, Adam, ecc.) sono i motori che alimentano la rivoluzione del deep learning. Questi algoritmi, pur essendo metodi di ottimizzazione locale relativamente semplici, quando applicati a reti neurali con miliardi di parametri e addestrati su enormi dataset, producono risultati straordinari: riconoscimento di immagini superiore a quello umano, traduzione automatica fluente, generazione di testo coerente, e molto altro.

Principi Unificanti

Attraverso tutti questi esempi, emergono principi unificanti:

1. Trade-off e compromessi: Raramente esiste una singola quantità da ottimizzare. Le soluzioni reali devono bilanciare obiettivi multipli. L'esagono dell'alveare ottimizza materiale ma deve anche fornire resistenza strutturale. Le reti vascolari minimizzano energia ma devono anche essere robuste a danni. Gli algoritmi di machine learning minimizzano errore sui dati di training ma devono generalizzare a dati nuovi.

2. Vincoli plasmano soluzioni: I vincoli non sono ostacoli ma elementi che definiscono il problema. La forma della catenaria emerge dal vincolo della lunghezza fissa. La filotassi aurea emerge dal vincolo di massimizzare luce solare evitando sovrapposizioni. Le architetture di reti neurali sono plasmate da vincoli computazionali e di memoria.

3. Ottimi locali vs globali: Molti problemi di ottimizzazione reali hanno paesaggi energetici complessi con molteplici ottimi locali. Metodi diversi navigano questo paesaggio in modi diversi: gradient descent cerca ottimi locali vicini, simulated annealing usa "temperatura" per saltare barriere, algoritmi genetici mantengono popolazione diversificata per esplorare multiple regioni.

4. Struttura gerarchica: Molti sistemi ottimali mostrano organizzazione gerarchica. Le reti vascolari hanno vasi di ordini diversi. Le espressioni evolute dalla programmazione genetica hanno sotto-espressioni riutilizzabili. Le reti neurali profonde hanno strati che estraggono feature a livelli di astrazione crescenti.

5. Robustezza attraverso ridondanza: Sistemi ottimali non sono fragili ma robusti. Le ragnatele hanno struttura ridondante che permette funzionamento anche con danni locali. Gli algoritmi evolutivi mantengono diversità per evitare convergenza prematura. Le reti neurali usano dropout per prevenire overfitting.

Prospettive Future

Il campo dell'ottimizzazione continua a evolversi rapidamente. Alcune direzioni promettenti:

Ottimizzazione quantistica: Computer quantistici potrebbero risolvere certi problemi di ottimizzazione esponenzialmente più velocemente di computer classici. Algoritmi come quantum annealing sono già implementati in hardware commerciale.

AutoML e Meta-learning: Invece di richiedere expertise umana per progettare modelli e scegliere iperparametri, sistemi automatici ottimizzano l'intero pipeline di machine learning.

Meta-learning va oltre: impara come imparare, ottimizzando strategie di ottimizzazione stesse.

Ottimizzazione multi-scala: Problemi che richiedono ottimizzazione simultanea a scale diverse - da atomica a macroscopica in materiali, da locale a globale in reti, da breve a lungo termine in pianificazione.

Ottimizzazione con apprendimento: Combinare ottimizzazione basata su modelli con apprendimento da dati. Per esempio, usare reinforcement learning per imparare euristiche per problemi combinatori, o usare reti neurali per approssimare soluzioni a PDEs in problemi variazionali.

Ottimizzazione etica e fairness: Non solo massimizzare accuracy o profitto, ma incorporare vincoli di fairness, privacy, interpretabilità. Ottimizzazione multi-obiettivo dove alcuni obiettivi rappresentano valori sociali.

Riflessione Finale

L'ottimizzazione è più di un insieme di tecniche matematiche o algoritmi computazionali. È una lente attraverso cui possiamo comprendere il mondo naturale, un principio che unifica fenomeni apparentemente disparati, e uno strumento potente per risolvere problemi pratici di crescente complessità. La bellezza dell'ottimizzazione risiede nella sua universalità. Gli stessi principi matematici appaiono nella fisica delle bolle di sapone, nella biologia della filotassi, nell'ingegneria dei ponti, nella logistica dei trasporti, nell'addestramento di reti neurali. Questa universalità non è accidentale ma riflette strutture profonde della matematica e della realtà fisica. Quando Fermat propose che la luce segue il percorso di tempo minimo, stava articolando più di una legge dell'ottica. Stava suggerendo che la natura stessa è un ottimizzatore, che i processi fisici "scegliono" configurazioni che estremizzano certe quantità. Questa intuizione è stata formalizzata nel principio di minima azione, che sottende non solo la meccanica classica ma anche la relatività e la meccanica quantistica. Quando Holland propose gli algoritmi genetici, non stava solo creando una tecnica computazionale, ma riconosceva che l'evoluzione naturale è un algoritmo di ottimizzazione che ha operato su scala planetaria per miliardi di anni, producendo la straordinaria diversità e complessità della vita. Quando i pionieri del deep learning applicarono il gradient descent a reti neurali sempre più grandi, non anticipavano probabilmente quanto sarebbe stato potente questo approccio apparentemente semplice. Eppure, l'ottimizzazione iterativa di miliardi di parametri ha prodotto sistemi che sfidano le nostre concezioni di cosa le macchine possano fare. In definitiva, lo studio dell'ottimizzazione ci insegna umiltà e meraviglia. Umiltà, perché ci ricorda che molte delle "soluzioni ingegnose" che scopriamo sono state già trovate dalla natura attraverso processi evolutivi. Meraviglia, perché rivela l'eleganza matematica sottostante a fenomeni naturali, e il potere di principi semplici applicati sistematicamente. Mentre affrontiamo sfide globali sempre più complesse - cambiamento climatico, pandemie, sostenibilità energetica, sistemi sanitari, educazione su scala - avremo bisogno di tutti gli strumenti di ottimizzazione a nostra disposizione. Dovremo combinare intuizioni dalla matematica pura, ispirazione dalla natura, potenza computazionale crescente, e saggezza per formulare i problemi giusti con gli obiettivi e vincoli appropriati. La natura ha avuto miliardi di anni per ottimizzare. Noi abbiamo solo decenni o secoli. Ma abbiamo un vantaggio: la capacità di comprendere i principi sottostanti, di formalizzarli

matematicamente, di implementarli computazionalmente, e di applicarli creativamente a nuovi domini. Questa è la promessa dell'ottimizzazione: non solo comprendere come la natura risolve problemi, ma imparare da essa per risolvere i nostri.

TEORIA DELLA COMPLESSITÀ COMPUTAZIONALE

Complessità dei Problemi di Ottimizzazione

Fondamenti: Classi P e NP

La teoria della complessità computazionale fornisce un framework formale per classificare i problemi in base alla difficoltà intrinseca di risolverli. Questo è cruciale per l'ottimizzazione, perché ci dice quali problemi possono essere risolti efficientemente e quali richiedono inevitabilmente approcci approssimati.

Definizioni rigorose

Un problema di decisione è una domanda con risposta sì/no. Per esempio:

- Problema di decisione del TSP: "Dato un grafo pesato G e un intero k , esiste un tour di lunghezza $\leq k$?"

La classe P (Polynomial time) contiene tutti i problemi di decisione risolvibili in tempo polinomiale, ovvero per cui esiste un algoritmo che su input di dimensione n termina in tempo $O(n^c)$ per qualche costante c .

La classe NP (Nondeterministic Polynomial time) contiene tutti i problemi di decisione per cui, data una soluzione candidata (un "certificato"), possiamo verificare in tempo polinomiale se è corretta.

Formalmente: un problema L appartiene a NP se esiste un verificatore polinomiale V tale che:

- x appartiene a L se e solo se esiste un certificato y con $|y| \leq p(|x|)$ tale che $V(x, y) = \text{"sì"}$
- V esegue in tempo polinomiale

Esempio: Per il TSP decisionale, il certificato è un ordine delle città. Possiamo verificare in tempo $O(n)$ se questo ordine forma un tour e in tempo $O(n)$ se la lunghezza è $\leq k$.

La relazione $P \subseteq NP$

È immediato che $P \subseteq NP$: se possiamo risolvere un problema in tempo polinomiale, possiamo anche verificare soluzioni in tempo polinomiale (basta risolvere da capo e confrontare).

Il problema del millennio: $P = NP$?

La domanda fondamentale, uno dei sette Problemi del Millennio del Clay Mathematics Institute con premio di un milione di dollari: è $P = NP$?

In altre parole: se possiamo verificare efficientemente le soluzioni, possiamo anche trovarle efficientemente?

L'intuizione comune è $P \neq NP$: verificare è più facile che trovare. Ma dopo 50 anni di ricerca intensiva, non abbiamo né una dimostrazione che $P = NP$ né che $P \neq NP$.

NP-completezza

Un problema L è NP-completo se:

1. L appartiene a NP
2. Ogni problema in NP può essere ridotto a L in tempo polinomiale

La seconda condizione significa che L è "almeno tanto difficile" quanto qualsiasi problema in NP. Se trovassimo un algoritmo polinomiale per un problema NP-completo, avremmo $P = NP$.

Il teorema di Cook-Levin (1971)

Il primo problema dimostrato NP-completo fu SAT (soddisfacibilità booleana):

Dato: formula booleana in forma normale congiuntiva (CNF) Domanda: esiste un assegnamento di verità che rende la formula vera?

Teorema (Cook-Levin): SAT è NP-completo.

Dimostrazione (sketch): Ogni computazione di una macchina di Turing non deterministica che esegue in tempo polinomiale può essere codificata come una formula SAT di dimensione polinomiale. La formula è soddisfacibile se e solo se la macchina accetta.

Riduzioni e la famiglia dei problemi NP-completi

Una volta stabilito che SAT è NP-completo, possiamo dimostrare che altri problemi lo sono tramite riduzioni:

Per mostrare che un problema L è NP-completo:

1. Mostra che L appartiene a NP (fornisci un verificatore polinomiale)
2. Riduci un problema già noto come NP-completo a L

Esempio: Riduzione da 3-SAT a Clique

3-SAT: formula CNF con esattamente 3 letterali per clausola Clique: dato grafo G e intero k , esiste un insieme di k vertici tutti connessi tra loro?

Riduzione: Data formula 3-SAT ϕ con m clausole, costruisci grafo G :

- Un vertice per ogni letterale in ogni clausola ($3m$ vertici totali)

- Connetti due vertici se:
 - Appartengono a clausole diverse
 - Non sono l'uno la negazione dell'altro

Affermazione: φ è soddisfacibile $\Leftrightarrow$ G ha una clique di dimensione m

Dimostrazione ($\rightarrow$): Se φ è soddisfacibile, ogni clausola ha almeno un letterale vero. Scegli un letterale vero per clausola. Questi m vertici formano una clique perché:

- Sono in clausole diverse (per costruzione dei vertici)
- Nessuno è negazione dell'altro (altrimenti l'assegnamento sarebbe inconsistente)

Dimostrazione ($\leftarrow$): Se G ha clique di dimensione m, deve contenere esattamente un vertice per clausola (non possono essere due dalla stessa clausola per costruzione). Questi vertici corrispondono a letterali non contraddittori. Assegna vero a questi letterali $\rightarrow$ tutte le m clausole sono soddisfatte.

Problemi di Ottimizzazione e NP-hardness

I problemi di ottimizzazione non sono problemi di decisione (chiedono il minimo/massimo, non sì/no). Definiamo:

Un problema di ottimizzazione è NP-hard se il corrispondente problema di decisione è NP-completo.

Esempio: TSP di ottimizzazione (trova il tour più breve) è NP-hard perché TSP decisionale (esiste tour di lunghezza $\leq k$?) è NP-completo.

Implicazioni pratiche

Se $P \neq NP$ (come quasi tutti credono), allora:

- Non esiste algoritmo polinomiale che risolve esattamente i problemi NP-hard
- Nel caso peggiore, dobbiamo esplorare un numero esponenziale di soluzioni
- Per istanze grandi, l'ottimizzazione esatta è computazionalmente impraticabile

Questo giustifica l'uso di metaeuristiche e algoritmi di approssimazione.

11.3 Teoria dell'Approssimazione

Se non possiamo trovare l'ottimo efficientemente, possiamo trovare soluzioni "vicine" all'ottimo?

Algoritmo di α -approssimazione

Per un problema di minimizzazione, un algoritmo è una α -approssimazione (con $\alpha \geq 1$) se per ogni istanza produce una soluzione di costo C soddisfacente:

$$C \leq \alpha \times \text{OPT}$$

dove OPT è il costo ottimo.

Il valore α è chiamato rapporto di approssimazione o fattore di approssimazione.

Esempio: Vertex Cover

Problema: dato grafo $G = (V, E)$, trova il più piccolo insieme $S \subseteq V$ tale che ogni arco ha almeno un estremo in S .

Questo problema è NP-hard. Algoritmo greedy di 2-approssimazione:

Algoritmo Greedy-VC(G):

$S = \emptyset$

$E' = E$

Mentre $E' \neq \emptyset$:

Scegli un arco arbitrario (u, v) da E'

Aggiungi u e v a S

Rimuovi da E' tutti gli archi incidenti a u o v

Ritorna S

Analisi del rapporto di approssimazione:

Sia S la soluzione prodotta, OPT la soluzione ottima.

Osservazione chiave: Gli archi scelti dall'algoritmo sono disgiunti (non condividono vertici). Sia M questo insieme di archi. Abbiamo $|S| = 2|M|$.

Ogni vertex cover (incluso OPT) deve coprire tutti gli archi in M . Poiché gli archi sono disgiunti, servono almeno $|M|$ vertici per coprirli. Quindi:

$$|OPT| \geq |M|$$

$$\text{Di conseguenza: } |S| = 2|M| \leq 2|OPT|$$

Quindi l'algoritmo è una 2-approssimazione.

Esempio: TSP Metrico

Per il TSP con disuguaglianza triangolare ($d(a, c) \leq d(a, b) + d(b, c)$), esiste un algoritmo di 2-approssimazione:

Algoritmo TSP-Approx(G):

Trova un Minimum Spanning Tree T di G

Duplica ogni arco di $T \rightarrow$ ottieni multigrafo euleriano H

Trova tour euleriano E in H

Converti E in tour hamiltoniano T eliminando vertici ripetuti (shortcutting)

Ritorna T

Analisi:

Sia OPT il tour ottimo, $cost(OPT)$ il suo costo.

Passo 1: Il MST ha costo $\leq cost(OPT)$ Dimostrazione: Se rimuoviamo un arco dal tour ottimo, otteniamo uno spanning tree. Quindi $MST \leq OPT - \text{un arco} < OPT$.

Passo 2: Il tour euleriano su H (grafo duplicato) ha costo $= 2 \times cost(MST) \leq 2 \times cost(OPT)$

Passo 3: Il shortcutting non aumenta il costo (per la disuguaglianza triangolare)

Quindi: $cost(T) \leq 2 \times cost(OPT)$

Teorema di Christofides (1976):

Esiste un algoritmo di $3/2$ -approssimazione per TSP metrico. È il miglior rapporto noto da quasi 50 anni!

L'algoritmo è più sofisticato:

1. Trova MST
2. Trova matching perfetto di peso minimo sui vertici di grado dispari del MST
3. Combina MST e matching $\rightarrow$ grafo euleriano
4. Shortcut del tour euleriano

Risultati di Non Approssimabilità

Per alcuni problemi, anche l'approssimazione è difficile.

Teorema PCP (Probabilistically Checkable Proofs)

Questo teorema profondo della teoria della complessità implica che per molti problemi NP-hard, non possono esistere buoni algoritmi di approssimazione (a meno che $P = NP$).

Gap di non approssimabilità per Max-3SAT

Max-3SAT: data formula 3-SAT, massimizza il numero di clausole soddisfatte.

Teorema (Håstad, 2001): Per ogni $\epsilon > 0$, è NP-hard distinguere tra:

- Istanze dove tutte le clausole possono essere soddisfatte (100%)
- Istanze dove al massimo $(7/8 + \epsilon)$ delle clausole possono essere soddisfatte

Corollario: Non esiste algoritmo di approssimazione con rapporto migliore di $8/7$ (a meno che $P = NP$).

Questo è notevole: un semplice algoritmo random (assegna vero/falso con probabilità $1/2$ a ogni variabile) soddisfa in media $7/8$ delle clausole! Quindi il random è quasi ottimale in senso di approssimazione.

Non Approssimabilità del TSP generale

Teorema: Il TSP senza disuguaglianza triangolare non ammette algoritmo di α -approssimazione per nessuna costante α (a meno che $P = NP$).

Dimostrazione (sketch): Riduzione da Hamiltonian Path (HP).

Dato grafo G , vogliamo decidere se ha un cammino hamiltoniano. Costruiamo istanza TSP:

- Vertici del TSP = vertici di G
- Distanze: $d(u, v) = 1$ se (u, v) è arco in G , $d(u, v) = n \times \alpha$ altrimenti

Se G ha cammino hamiltoniano, $OPT_TSP = n$ (usa gli archi di G) Se G non ha cammino hamiltoniano, ogni tour usa almeno un "arco lungo", quindi $cost \geq n \times \alpha$

Se avessimo α -approssimazione per TSP, potremmo distinguere questi casi risolvendo HP in tempo polinomiale $\rightarrow P = NP$.

Classi di Complessità Avanzate

Classe coNP

Un problema L appartiene a coNP se il suo complemento appartiene a NP.

Esempio: Tautologia (formula booleana sempre vera?) appartiene a coNP perché il suo complemento (formula non tautologia = esiste assegnamento che la falsifica) appartiene a NP.

È un problema aperto se $NP = coNP$. Si congettura $NP \neq coNP$.

Gerarchia Polinomiale

Definita induttivamente:

- $\Sigma_0^P = \Pi_0^P = P$
- $\Sigma_{i+1}^P = NP^{+kP}$ (problemi risolvibili in tempo polinomiale con oracolo per problemi in Σ_i^P)
- $\Pi_{i+1}^P = coNP^{+kP}$

Quindi:

- $\Sigma_1^P = NP$
- $\Pi_1^P = coNP$

- $\Sigma_2^P = \text{NP}^{\text{NP}}$ (problemi risolvibili con oracolo NP)

Esempio di problema Σ_2^P -completo: "Esiste una formula booleana tale che per ogni assegnamento delle variabili x , esiste un assegnamento delle variabili y che soddisfa la formula?"

Questo ha quantificatori alternati: $\exists x \forall y \varphi(x, y)$

NP: Counting Complexity

NP è la classe dei problemi di conteggio. Un problema appartiene a NP se conta il numero di soluzioni di un problema NP.

Esempio: NSAT conta il numero di assegnamenti soddisfacenti di una formula booleana.

Teorema (Valiant): NSAT è NP-completo.

Fatto sorprendente: NP sembra più difficile di NP. Calcolare il permanente di una matrice è NP-completo, mentre decidere se è non-zero è in P!

Applicazione all'ottimizzazione

Molti problemi di ottimizzazione combinatoria coinvolgono conteggio:

- Quante soluzioni ottime esistono?
- Qual è la probabilità che una soluzione casuale sia vicina all'ottimo?

Questi problemi sono tipicamente #P-completi, indicando che anche ottenere informazioni statistiche sull'ottimo è computazionalmente difficile.

Complessità Parametrizzata

L'idea: invece di classificare problemi solo per dimensione input n , consideriamo anche un parametro k .

Un problema è Fixed-Parameter Tractable (FPT) rispetto a k se può essere risolto in tempo $O(f(k) \times p(n))$ dove f è una funzione arbitraria e p è un polinomio.

Esempio: Vertex Cover parametrizzato

Input: Grafo G , intero k Domanda: Esiste vertex cover di dimensione $\leq k$? Parametro: k

Algoritmo FPT:

Bounded-Search-Tree(G, k):

Se $k < 0$: ritorna NO

Se $E = \emptyset$: ritorna SI

Scegli un arco (u, v)

Se $\text{Bounded-Search-Tree}(G - u, k - 1) = \text{Sì}$: ritorna Sì

Se $\text{Bounded-Search-Tree}(G - v, k - 1) = \text{Sì}$: ritorna Sì

Ritorna NO

Analisi: Ad ogni chiamata ricorsiva, k diminuisce di 1. Ci sono al massimo 2^k chiamate. Ogni chiamata richiede tempo $O(n + m)$. Totale: $O(2^k \times (n + m))$

Questo è FPT perché esponenziale solo in k , non in n . Per k piccolo (vertex cover di dimensione ≤ 20), è pratico anche per grafi enormi.

W-gerarchia

Non tutti i problemi NP sono FPT. La W-gerarchia classifica i problemi parametrizzati:

$$\text{FPT} \subseteq W[1] \subseteq W[2] \subseteq \dots \subseteq \text{XP}$$

Si congettura che queste inclusioni siano strette. Un problema $W[1]$ -hard probabilmente non è FPT.

Esempio: Clique parametrizzato per dimensione k è $W[1]$ -completo. Non si conosce algoritmo FPT.

Implicazioni pratiche

La complessità parametrizzata guida la progettazione di algoritmi:

- Se il problema è FPT rispetto a un parametro naturalmente piccolo, vale la pena progettare algoritmi esatti efficienti
- Altrimenti, meglio focalizzarsi su approssimazione o euristiche

Esempi di parametri utili:

- Treewidth del grafo (molti problemi NP-hard sono FPT rispetto a treewidth)
- Numero di variabili in formule booleane
- Numero di colori in problemi di colorazione

Complessità in Media vs. Caso Peggior

Molti problemi NP-hard sono difficili solo nel caso peggiore. In pratica, istanze casuali possono essere facili.

Smoothed Analysis

Proposta da Spielman e Teng per analizzare il Simplex per programmazione lineare.

Idea: Perturbiamo leggermente l'input con rumore casuale. Quanto è difficile il problema perturbato in media?

Questo interpola tra:

- Caso peggiore (nessuna perturbazione)
- Caso medio (perturbazione totale = distribuzione uniforme)

Per molti algoritmi pratici (incluso Simplex, algoritmi per TSP, ecc.), la smoothed complexity è polinomiale anche se la worst-case complexity è esponenziale.

Instance-specific bounds

Per problemi come TSP, possiamo caratterizzare la difficoltà di istanze specifiche:

- Istanze con forte struttura geometrica (città su griglia) sono più facili
- Istanze random uniformi hanno proprietà probabilistiche sfruttabili
- Istanze "worst-case" costruite apposta sono rare in pratica

Questo giustifica l'efficacia di euristiche su istanze reali anche per problemi teoricamente intrattabili.

Proprietà tipiche vs. proprietà peggiori

Esempio: 3-SAT casuale

Generazione: n variabili, m clausole, ogni clausola sceglie 3 variabili uniformly e le nega con probabilità $1/2$.

Transizione di fase: Definiamo $\alpha = m/n$ (rapporto clausole/variabili)

- $\alpha < 4.26$: con alta probabilità, la formula è soddisfacibile e algoritmi semplici trovano soluzione rapidamente
- $\alpha > 4.26$: con alta probabilità, la formula è insoddisfacibile
- $\alpha \approx 4.26$: regione critica, istanze più difficili

Nella regione critica, anche i migliori SAT solver moderni faticano su istanze con $n \approx 1000$.

Ma la maggior parte delle istanze reali (da verifica hardware, planning, ecc.) hanno struttura diversa e sono più trattabili.

Conclusioni e Frontiere

La teoria della complessità fornisce un quadro per capire quando l'ottimizzazione esatta è fattibile e quando dobbiamo accontentarci di approssimazioni o euristiche.

Lezioni chiave:

1. Molti problemi di ottimizzazione sono NP-hard $\rightarrow$ nessun algoritmo polinomiale esatto (se $P \neq NP$)
2. Per alcuni problemi esistono buoni algoritmi di approssimazione (Vertex Cover: 2-approx, TSP metrico: 3/2-approx)

3. Per altri, anche l'approssimazione è hard (TSP generale: non approssimabile)
4. La complessità parametrizzata offre una via intermedia: esatto ed efficiente quando certi parametri sono piccoli
5. Il caso peggiore non sempre riflette la difficoltà pratica

Problemi aperti importanti:

- P vs NP: il problema centrale
- Può TSP metrico essere approssimato meglio di $3/2$? (aperto da 50 anni!)
- Esistono problemi in $NP \setminus P$ che non sono NP-completi? (se $P \neq NP$, sì per teorema di Ladner, ma nessun candidato naturale è conosciuto)
- La gerarchia polinomiale collassa? (si crede di no ma non dimostrato)

Comprendere queste limitazioni teoriche è essenziale per chi progetta algoritmi di ottimizzazione: ci dice quando cercare soluzioni esatte, quando accontentarsi di approssimazioni garantite, e quando ricorrere a euristiche senza garanzie ma efficaci in pratica.

OTTIMIZZAZIONE MULTI-OBIETTIVO

Ottimizzazione con Obiettivi Multipli in Conflitto

Fondamenti della Ottimizzazione Multi-Obiettivo

Nella maggior parte dei problemi reali di ottimizzazione, dobbiamo bilanciare obiettivi multipli che sono spesso in conflitto. Un ingegnere che progetta un'automobile vuole massimizzare la sicurezza, minimizzare il costo, massimizzare l'efficienza energetica, minimizzare le emissioni, e massimizzare il comfort. Questi obiettivi non possono essere simultaneamente ottimizzati: una macchina più sicura tipicamente costa di più, un motore più efficiente potrebbe essere meno potente, e così via.

Formulazione matematica

Un problema di ottimizzazione multi-obiettivo (MOP) è definito come:

minimizza $F(x) = (f_1(x), f_2(x), \dots, f_k(x))$ soggetto a $x \in X$

dove:

- x è il vettore decisionale (le variabili che controlliamo)
- X è l'insieme delle soluzioni ammissibili (definito da vincoli)
- $f_i : X \rightarrow \mathbb{R}$ sono le k funzioni obiettivo
- $F : X \rightarrow \mathbb{R}^k$ è il vettore delle funzioni obiettivo

A differenza dell'ottimizzazione single-obiettivo, non esiste generalmente una singola soluzione ottimale, ma un insieme di soluzioni di compromesso.

Esempio concreto: progettazione di un ponte

Consideriamo la progettazione di un ponte sospeso. Vogliamo:

- $f_1(x)$: minimizzare il costo di costruzione
- $f_2(x)$: minimizzare il peso (per ridurre carico su fondazioni)
- $f_3(x)$: massimizzare la capacità di carico
- $f_4(x)$: minimizzare il tempo di costruzione

dove x include parametri come: lunghezza dei cavi, spessore della carreggiata, numero di piloni, tipo di materiali, ecc.

Questi obiettivi sono chiaramente in conflitto:

- Aumentare la capacità di carico richiede materiali più resistenti → aumenta costo
- Ridurre il peso può richiedere materiali più costosi (leghe leggere)
- Ridurre il tempo richiede più manodopera → aumenta costo

Dominanza di Pareto

Il concetto centrale nell'ottimizzazione multi-obiettivo è la dominanza di Pareto, introdotta dall'economista Vilfredo Pareto nel 1896.

Definizione formale

Una soluzione x_1 domina (in senso di Pareto) una soluzione x_2 , denotato $x_1 < x_2$, se:

1. $f_i(x_1) \leq f_i(x_2)$ per ogni $i \in \{1, 2, \dots, k\}$
2. Esiste almeno un j tale che $f_j(x_1) < f_j(x_2)$

In altre parole, x_1 domina x_2 se è almeno altrettanto buona in tutti gli obiettivi e strettamente migliore in almeno un obiettivo.

Interpretazione geometrica

Nello spazio degli obiettivi $\mathbb{R}^k$, il punto $F(x_1) = (f_1(x_1), \dots, f_k(x_1))$ domina $F(x_2)$ se si trova "a sud-ovest" di $F(x_2)$ (assumendo minimizzazione): ogni coordinata è $\leq$ e almeno una è strettamente $<$.

Non-dominanza e ottimalità di Pareto

Una soluzione x^* è detta Pareto-ottimale (o non-dominata) se non esiste nessuna soluzione $x \in X$ che la domina.

Formalmente: x^* è Pareto-ottimale $\Leftrightarrow \neg \exists x \in X : x < x^*$

L'insieme di tutte le soluzioni Pareto-ottimali è chiamato insieme Pareto-ottimale (Pareto Set) nello spazio decisionale, e la sua immagine nello spazio obiettivo è il fronte di Pareto (Pareto Front).

Proprietà del fronte di Pareto

1. **Non è un singolo punto** (a meno di casi degeneri): il fronte contiene tipicamente infinite soluzioni per problemi continui
2. **Rappresenta i trade-off ottimali**: muoversi lungo il fronte significa migliorare un obiettivo peggiorando altri
3. **Forma una superficie (k-1)-dimensionale** in $\mathbb{R}^k$: per $k=2$ è una curva, per $k=3$ è una superficie 2D

Esempio numerico

Consideriamo un problema bi-obiettivo semplice:

minimizza $f_1(x) = x^2$ minimizza $f_2(x) = (x - 2)^2$ con $x \in [0, 3]$

Analisi:

- f_1 è minimizzata per $x = 0 \rightarrow f_1(0) = 0, f_2(0) = 4$
- f_2 è minimizzata per $x = 2 \rightarrow f_1(2) = 4, f_2(2) = 0$
- Ogni punto $x \in [0, 2]$ è Pareto-ottimale!

Verifica: per $x \in (0, 2)$, se aumentiamo x miglioriamo f_2 ma peggioriamo f_1 ; se diminuiamo x miglioriamo f_1 ma peggioriamo f_2 . Nessun punto domina gli altri.

Il fronte di Pareto nello spazio obiettivo è la curva: $\{(x^2, (x-2)^2) : x \in [0, 2]\} = \{(t, 4 - 4\sqrt{t} + t) : t \in [0, 4]\}$

Classificazione delle Soluzioni

Oltre alle soluzioni Pareto-ottimali, è utile classificare le altre soluzioni.

Dominanza debole

x_1 domina debolmente x_2 ($x_1 \leq x_2$) se $f_i(x_1) \leq f_i(x_2)$ per ogni i (senza richiedere disuguaglianza stretta in almeno un obiettivo).

Fronti di Pareto multipli

Possiamo stratificare le soluzioni in fronti multipli:

- Fronte 1: soluzioni non-dominate dall'intero insieme
- Fronte 2: soluzioni non-dominate dopo rimozione del Fronte 1
- Fronte 3: soluzioni non-dominate dopo rimozione dei Fronti 1 e 2

- ...

Questa stratificazione è usata in algoritmi come NSGA-II.

Procedura di non-dominated sorting

Input: Insieme di soluzioni S

Output: Fronti $F_1, F_2, \dots, F_m$

$F_1 = \{x \in S : x \text{ non è dominata da nessun'altra soluzione in } S\}$

$i = 1$

Mentre $S \setminus (F_1 \cup \dots \cup F_i) \neq \emptyset$:

$S' = S \setminus (F_1 \cup \dots \cup F_i)$

$F_{i+1} = \{x \in S' : x \text{ non è dominata da nessuna soluzione in } S'\}$

$i = i + 1$

Complessità: $O(k \times N^2)$ dove $N = |S|$ e k è il numero di obiettivi, usando l'algoritmo naive. Esistono algoritmi più efficienti: $O(N \log^{k-1} N)$ per k obiettivi.

Metriche di Qualità per Insiemi di Pareto

Come valutiamo la qualità di un insieme approssimato di soluzioni Pareto-ottimali?

Metriche principali

1. Convergenza (Generational Distance)

Misura quanto l'insieme approssimato A è vicino al vero fronte di Pareto P^* :

$$GD(A, P^*) = (1/|A|) \times \sum_{a \in A} d(a, P^*)$$

dove $d(a, P^*) = \min_{p^* \in P^*} \|F(a) - F(p^*)\|$ è la distanza minima da a al fronte vero.

Interpretazione: $GD \rightarrow 0$ significa che l'insieme approssimato converge al fronte vero.

2. Diversità (Spacing)

Misura quanto uniformemente le soluzioni sono distribuite:

$$S(A) = \sqrt{[(1/|A|) \times \sum_{a \in A} (\bar{d} - d_a)^2]}$$

dove:

- $d_a = \min_{a' \in A, a' \neq a} \|F(a) - F(a')\|$ è la distanza dal vicino più prossimo
- $\bar{d} = (1/|A|) \times \sum_{a \in A} d_a$ è la distanza media

Interpretazione: $S \rightarrow 0$ significa distribuzione perfettamente uniforme.

3. Estensione (Spread)

Misura quanto ampiamente l'insieme copre il fronte:

$$\Delta = (d_f + d_l + \sum_{i=1}^{|A|-1} |d_i - \bar{d}|) / (d_f + d_l + (|A| - 1) \times \bar{d})$$

dove d_f e d_l sono le distanze degli estremi del fronte approssimato dagli estremi del fronte vero.

4. Ipervolume (Hypervolume Indicator)

La metrica più importante e ampiamente usata:

$HV(A)$ = volume dello spazio dominato da A rispetto a un punto di riferimento r

Formalmente, per un problema di minimizzazione:

$$HV(A) = \text{volume}(\{z \in \mathbb{R}^k : \exists a \in A : F(a) \leq z \leq r\})$$

dove $\leq$ è componentwise e r è un punto di riferimento peggiore di tutti i punti nel fronte.

Proprietà dell'ipervolume:

1. **Monotonia di Pareto:** se $A < B$ (ogni punto in A domina almeno un punto in B), allora $HV(A) > HV(B)$
2. **Sensibilità alla convergenza e diversità:** cattura sia quanto vicino al fronte vero che quanto ben distribuito
3. **Unica metrica Pareto-conforme:** è l'unica metrica (a meno di monotonia) che garantisce che migliorare l'ipervolume implica avvicinarsi al fronte Pareto

Calcolo dell'ipervolume

Per $k = 2$ (due obiettivi), il calcolo è semplice: $O(N \log N)$ dove $N = |A|$.

Per $k = 3$, esistono algoritmi $O(N^2 \log N)$.

Per $k \geq 4$, il problema diventa #P-hard nel caso peggiore! Esistono algoritmi pratici ma esponenziali in k .

Algoritmo per $k = 2$:

Hypervolume-2D(A, r):

Ordina A per f_1 crescente: $a_1, a_2, \dots, a_N$

hv = 0

$y_{\text{prev}} = r_2$

Per i da 1 a N :

$$h_v = h_v + (r_1 - f_1(a_i)) \times (y_{\text{prev}} - f_2(a_i))$$

$$y_{\text{prev}} = f_2(a_i)$$

Ritorna h_v

Algoritmi per Ottimizzazione Multi-Obiettivo

Categorie di approcci

1. **Scalarizzazione:** Combina obiettivi multipli in uno singolo
2. **A posteriori:** Genera l'intero fronte di Pareto, decisione fatta dopo
3. **A priori:** Decision maker specifica preferenze prima, algoritmo cerca in quella regione
4. **Interattivo:** Alternanza tra ottimizzazione e feedback dal decision maker

Metodo dei pesi (Weighted Sum)

L'approccio più semplice: combina obiettivi con pesi:

minimizza $\sum_{i=1}^k w_i \times f_i(x)$ soggetto a $x \in X$, $w_i \geq 0$, $\sum_i w_i = 1$

Teorema: Se x^* minimizza $\sum_i w_i \times f_i(x)$ con $w_i > 0$ per ogni i , allora x^* è Pareto-ottimale.

Dimostrazione: Per contraddizione, supponiamo x^* non sia Pareto-ottimale. Allora esiste $y \in X$ che domina x^* , ovvero $f_i(y) \leq f_i(x^*)$ per ogni i con almeno una disuguaglianza stretta. Moltiplicando per $w_i > 0$ e sommando:

$$\sum_i w_i f_i(y) < \sum_i w_i f_i(x^*)$$

Contraddizione con x^* che minimizza la somma pesata.

Limitazione: Il metodo dei pesi NON può trovare soluzioni nel fronte di Pareto non-convesso.

Esempio: Se il fronte ha una regione concava, nessuna scelta di pesi produce soluzioni in quella regione. Questo è un problema serio in pratica.

ϵ -Constraint Method

Ottimizza un obiettivo principale mentre vincola gli altri:

minimizza $f_j(x)$ soggetto a $f_i(x) \leq \epsilon_i$ per $i \neq j$, $x \in X$

Variando gli ϵ_i , generiamo diverse soluzioni Pareto-ottimali.

Vantaggio: Può trovare soluzioni in fronti non-convessi.

Svantaggio: Richiede molte esecuzioni con parametri diversi; difficile scegliere gli ϵ_i appropriati.

NSGA-II: Non-dominated Sorting Genetic Algorithm II

NSGA-II, proposto da Deb et al. nel 2002, è l'algoritmo multi-obiettivo evolutivo più influente.

Componenti chiave

1. **Non-dominated sorting:** Classifica popolazione in fronti
2. **Crowding distance:** Misura di densità per preservare diversità
3. **Selezione elitaria:** Combina genitori e figli, seleziona i migliori

Crowding Distance

Per ogni soluzione i nel fronte F , calcola:

$$CD_i = \sum_{m=1}^k [f_m(i+1) - f_m(i-1)] / [f_m^{\max} - f_m^{\min}]$$

dove $i-1$ e $i+1$ sono i vicini di i quando le soluzioni in F sono ordinate per f_m .

Interpretazione: CD_i misura il perimetro del cuboide che ha i come vertice e i suoi vicini come lati opposti. Valori alti = regione meno affollata.

Algoritmo NSGA-II

NSGA-II(N_{pop} , N_{gen}):

P_0 = Inizializza popolazione casuale di dimensione N_{pop}

Valuta $F(x)$ per ogni x in P_0

Per t da 0 a $N_{gen} - 1$:

Q_t = Genera figli da P_t usando crossover e mutazione

Valuta $F(x)$ per ogni x in Q_t

$R_t = P_t \cup Q_t$ (combina genitori e figli, dimensione $2N_{pop}$)

$(F_1, F_2, \dots, F_m) = \text{Non-dominated-sort}(R_t)$

$P_{t+1} = \emptyset$

$i = 1$

Mentre $|P_{t+1}| + |F_i| \leq N_{pop}$:

Calcola crowding-distance per soluzioni in F_i

$$P_{\{t+1\}} = P_{\{t+1\}} \cup F_i$$

$$i = i + 1$$

Se $|P_{\{t+1\}}| < N_{\text{pop}}$:

Ordina F_i per crowding-distance decrescente

$$P_{\{t+1\}} = P_{\{t+1\}} \cup F_{i[1 : N_{\text{pop}} - |P_{\{t+1\}}|]}$$

Ritorna Fronte-1 di $P_{\{N_{\text{gen}}\}}$

Complessità: $O(k \times N^2 \times N_{\text{gen}})$ dove k = numero obiettivi, N = dimensione popolazione.

Perché NSGA-II funziona bene

1. **Pressione verso il fronte:** Non-dominated sorting favorisce soluzioni migliori
2. **Preservazione della diversità:** Crowding distance distribuisce soluzioni uniformemente
3. **Elitismo:** Non perde mai le migliori soluzioni trovate
4. **Scalabilità:** Ragionevole fino a $k \approx 10$ obiettivi

Limitazioni

Per $k > 10$ (many-objective optimization), NSGA-II degrada:

- Troppi punti diventano non-dominati $\rightarrow$ pressione selettiva debole
- Crowding distance in molte dimensioni è meno informativa

Algoritmi Moderni: NSGA-III e MOEA/D

NSGA-III (2014)

Estensione di NSGA-II per many-objective problems.

Idea chiave: Usa punti di riferimento ben distribuiti invece di crowding distance.

Generazione di punti di riferimento: Su un simplex $(k-1)$ -dimensionale, genera punti uniformemente spazati usando metodo di Das-Dennis.

Per $k = 3$, p divisioni per asse: $C(p + k - 1, k - 1) = C(p + 2, 2)$ punti di riferimento.

Selezione in NSGA-III:

1. Come NSGA-II, riempi $P_{\{t+1\}}$ con fronti $F_1, F_2, \dots$
2. Per l'ultimo fronte F_i da includere parzialmente:
 - Associa ogni soluzione al punto di riferimento più vicino
 - Conta quante soluzioni già in $P_{\{t+1\}}$ sono associate a ciascun punto
 - Privilegia punti di riferimento con meno soluzioni associate (nicchia preservation)

MOEA/D (Multi-Objective Evolutionary Algorithm based on Decomposition)

Approccio radicalmente diverso: decompone il problema multi-obiettivo in N sotto-problemi single-obiettivo.

Per ogni peso $\lambda^i = (\lambda_1^i, \dots, \lambda_k^i)$:

minimizza $g(x | \lambda^i) = \max_{j=1, \dots, k} \{ \lambda_j^i \times [f_j(x) - z_j^*] \}$

dove $z^* = (z_1^*, \dots, z_k^*)$ è il punto ideale (minimo di ogni obiettivo ignorando gli altri).

Questa è la Tchebycheff scalarization, che può catturare fronti non-convessi.

Algoritmo MOEA/D

MOEA/D(N):

Genera N vettori peso uniformemente distribuiti: $\lambda^1, \dots, \lambda^N$

Definisci vicinato per ogni λ^i : T vicini più simili (per distanza euclidea)

Inizializza popolazione $x^1, \dots, x^N$

Inizializza $z^* = (\min f_1, \min f_2, \dots, \min f_k)$

Mentre criterio di terminazione non soddisfatto:

Per i da 1 a N:

Seleziona indici casuali j, k dal vicinato di i

Genera figlio y da x^j e x^k usando crossover e mutazione

Aggiorna z^* : $z_m^* = \min(z_m^*, f_m(y))$ per ogni m

Per ogni indice j nel vicinato di i:

Se $g(y | \lambda^j) < g(x^j | \lambda^j)$:

$x^j = y$

Ritorna $\{x^1, \dots, x^N\}$

Vantaggi di MOEA/D:

- Scala meglio con numero di obiettivi (meno valutazioni per convergenza)
- Naturalmente parallelo (sotto-problemi indipendenti)
- Può usare conoscenza specifica del dominio per ogni sotto-problema

Applicazioni Reali

1. Progettazione ingegneristica: traliccio strutturale

Obiettivi:

- f_1 : Minimizzare peso totale
- f_2 : Minimizzare spostamento massimo sotto carico
- f_3 : Minimizzare costo

Variabili: Spessore, materiale, geometria delle aste

Risultato tipico: Fronte di Pareto con centinaia di configurazioni. L'ingegnere sceglie in base a vincoli non formalizzabili (estetica, disponibilità materiali, regolamentazioni).

2. Pianificazione di trattamenti medici

Esempio: Radioterapia per tumori

Obiettivi:

- f_1 : Massimizzare dose al tumore
- f_2 : Minimizzare dose ai tessuti sani circostanti
- f_3 : Minimizzare tempo totale trattamento

Il trade-off è critico: trattamenti più aggressivi sono più efficaci ma più dannosi. Il fronte di Pareto fornisce al medico opzioni chiare per decidere con il paziente.

3. Gestione di portafogli finanziari

Obiettivi:

- f_1 : Massimizzare rendimento atteso
- f_2 : Minimizzare rischio (varianza)
- f_3 : Minimizzare costo transazioni
- f_4 : Massimizzare liquidità

Questo è il classico problema di Markowitz esteso. L'investitore sceglie un punto sul fronte basandosi sulla propria tolleranza al rischio.

4. Design di automobili

Obiettivi multipli:

- Sicurezza (crash test scores)
- Efficienza carburante
- Prestazioni (accelerazione, velocità massima)
- Costo produzione
- Emissioni
- Comfort
- Estetica (spesso modellata da survey)

Risultato: Decine di configurazioni Pareto-ottimali. Marketing e management scelgono quali produrre in base a segmenti di mercato.

5. Pianificazione urbana

Esempio: Posizionamento di infrastrutture (scuole, ospedali, parchi)

Obiettivi:

- Minimizzare distanza media dalla popolazione
- Massimizzare copertura (% popolazione entro raggio)
- Minimizzare costo totale
- Minimizzare impatto ambientale
- Massimizzare equità (distribuire servizi uniformemente tra quartieri)

Il fronte di Pareto informa decisioni politiche, mostrando chiaramente i trade-off (es: coprire tutti costa X% in più ma riduce equità).

Direzioni di Ricerca Avanzate

Many-Objective Optimization ($k > 3$)

Sfide:

- Dominanza diventa rara: in alta dimensione, la maggior parte dei punti è non-dominata
- Visualizzazione impossibile
- Iper-volume computazionalmente proibitivo

Approcci promettenti:

- Grid-based methods: Partiziona spazio obiettivo in iper-celle
- Indicatore-based: Usa metriche diverse da dominanza (es: R2, IGD+)
- Preference-based: Incorpora preferenze decision maker per focalizzare ricerca

Dynamic Multi-Objective Optimization

Obiettivi cambiano nel tempo: $f_i(x, t)$

Esempio: Pianificazione traffico urbano dove domanda varia durante il giorno

Algoritmi devono:

- Rilevare cambiamenti nell'ambiente
- Adattare fronte Pareto rapidamente
- Predire evoluzioni future

Robust Multi-Objective Optimization

Parametri o obiettivi incerti. Cercare soluzioni che siano Pareto-ottimali e robuste a variazioni.

Formulazione tipica: per ogni x , $f_i(x)$ è una distribuzione.

Obiettivo: Minimizzare $E[f_i(x)]$ (valore atteso) e $\text{Var}[f_i(x)]$ (varianza), simultaneamente.

Multi-Objective Optimization con Machine Learning

Trend recente: Usare modelli surrogati (Gaussian Processes, Neural Networks) per approssimare funzioni obiettivo costose.

Bayesian Multi-Objective Optimization:

1. Costruisci modello probabilistico di $f_1, \dots, f_k$
2. Usa acquisition function multi-obiettivo (es: Expected Hypervolume Improvement)
3. Valuta punto più promettente
4. Aggiorna modello
5. Ripeti

Riduce drasticamente il numero di valutazioni necessarie (critico quando f_i richiede simulazioni costose).

Conclusioni

L'ottimizzazione multi-obiettivo riconosce che la realtà è intrinsecamente complessa e richiede bilanciamenti. Non cerca una soluzione ottimale ma un insieme di compromessi ottimali, lasciando al decision maker la scelta finale basata su preferenze, vincoli e informazioni non formalizzabili.

Gli algoritmi evolutivi multi-obiettivo (NSGA-II, NSGA-III, MOEA/D) sono diventati strumenti standard in ingegneria, medicina, finanza, e molti altri campi, permettendo di esplorare sistematicamente gli spazi dei trade-off e prendere decisioni informate in presenza di obiettivi contrastanti.

CONNESSIONI CON LA TEORIA DELL'INFORMAZIONE

Ottimizzazione, Informazione ed Entropia

Il Principio di Massima Entropia

La teoria dell'informazione, fondata da Claude Shannon nel 1948, e l'ottimizzazione sono profondamente connesse attraverso il concetto di entropia. Il principio di massima entropia fornisce un framework razionale per fare inferenze quando abbiamo informazione incompleta.

Entropia di Shannon

Data una distribuzione di probabilità discreta $p = (p_1, p_2, \dots, p_n)$ dove $p_i \geq 0$ e $\sum_i p_i = 1$, l'entropia di Shannon è definita come:

$$H(p) = -\sum_{i=1}^n p_i \log p_i$$

(con la convenzione $0 \log 0 = 0$). L'entropia misura l'incertezza media della distribuzione. In termini di teoria dell'informazione, è il numero medio di bit necessari per codificare un simbolo dalla distribuzione.

Proprietà fondamentali

1. **Non-negatività:** $H(p) \geq 0$ con uguaglianza se e solo se p è deterministica ($p_i = 1$ per qualche i)
2. **Massimalità:** $H(p) \leq \log n$ con uguaglianza se e solo se p è uniforme ($p_i = 1/n$ per ogni i)
3. **Concavità:** H è una funzione strettamente concava di p

Dimostrazione della massimalità

Vogliamo mostrare che $H(p) \leq \log n$. Usiamo la disuguaglianza di Jensen.

Consideriamo la distribuzione uniforme u con $u_i = 1/n$. La divergenza di Kullback-Leibler tra p e u è:

$$KL(p \parallel u) = \sum_i p_i \log(p_i / u_i) = \sum_i p_i \log(n p_i) = \sum_i p_i \log n + \sum_i p_i \log p_i = \log n - H(p)$$

Poiché $KL(p \parallel u) \geq 0$ con uguaglianza se e solo se $p = u$ (questo è un teorema fondamentale della teoria dell'informazione), abbiamo:

$$\log n - H(p) \geq 0 \rightarrow H(p) \leq \log n$$

con uguaglianza se e solo se $p_i = 1/n$ per ogni i .

Il principio di massima entropia come ottimizzazione vincolata

Il principio di massima entropia (MaxEnt) di Jaynes afferma: data informazione parziale su una distribuzione, espressa come vincoli sulla media di certe funzioni, la distribuzione "meno informativa" compatibile con i vincoli è quella che massimizza l'entropia.

Formulazione matematica:

massimizza $H(p) = -\sum_i p_i \log p_i$ soggetto a $E_p[f_k] = \sum_i p_i f_k(i) = c_k$ per $k = 1, \dots, m$
 $\sum_i p_i = 1$ $p_i \geq 0$ per ogni i

dove f_k sono funzioni note e c_k sono valori osservati (vincoli).

Soluzione via moltiplicatori di Lagrange

Formiamo la lagrangiana:

$$L(p, \lambda, \mu) = -\sum_i p_i \log p_i + \sum_k \lambda_k (\sum_i p_i f_k(i) - c_k) + \mu (\sum_i p_i - 1)$$

Derivando rispetto a p_i e ponendo uguale a zero:

$$\partial L / \partial p_i = -\log p_i - 1 + \sum_k \lambda_k f_k(i) + \mu = 0$$

Risolvendo per p_i :

$$p_i = \exp(\sum_k \lambda_k f_k(i) + \mu - 1) = Z^{-1} \exp(\sum_k \lambda_k f_k(i))$$

dove $Z = \exp(1 - \mu)$ è una costante di normalizzazione (funzione di partizione):

$$Z = \sum_i \exp(\sum_k \lambda_k f_k(i))$$

Questa è la forma esponenziale della distribuzione MaxEnt.

Esempi fondamentali

1. Nessun vincolo

Se non abbiamo vincoli (oltre alla normalizzazione), la MaxEnt è la distribuzione uniforme:

$$p_i = 1/n \text{ per ogni } i$$

2. Vincolo sulla media

Supponiamo di conoscere solo la media: $E[X] = \mu$ dove X assume valori $\{1, 2, \dots, n\}$.

$$\text{Il vincolo è: } \sum_{i=1}^n i \times p_i = \mu$$

La distribuzione MaxEnt è:

$$p_i = Z^{-1} \exp(\lambda i)$$

Determinando λ dal vincolo, questa è una distribuzione esponenziale discreta.

3. Vincolo su media e varianza (caso continuo)

Per variabile continua $X \in \mathbb{R}$ con $E[X] = \mu$ e $\text{Var}[X] = \sigma^2$, l'entropia differenziale è:

$$h(p) = -\int_{-\infty}^{\infty} p(x) \log p(x) dx$$

La distribuzione MaxEnt soggetta a questi vincoli è la gaussiana:

$$p(x) = (1/\sqrt{2\pi\sigma^2}) \exp(-(x - \mu)^2 / (2\sigma^2))$$

Dimostrazione:

Lagrangiana: $L = -\int p \log p \, dx + \lambda_1 \left(\int p \, dx - 1 \right) + \lambda_2 \left(\int x p \, dx - \mu \right) + \lambda_3 \left(\int (x - \mu)^2 p \, dx - \sigma^2 \right)$

Calcolo variazionale: $\delta L / \delta p = -\log p - 1 + \lambda_1 + \lambda_2 x + \lambda_3 (x - \mu)^2 = 0$

Quindi: $\log p = -1 + \lambda_1 + \lambda_2 x + \lambda_3 (x - \mu)^2$

Imponendo i vincoli, troviamo $\lambda_1, \lambda_2, \lambda_3$ e otteniamo $p(x)$ gaussiana.

Questo spiega perché la distribuzione gaussiana appare così frequentemente: è la distribuzione meno informativa data media e varianza.

Entropia Relativa e Proiezioni Informazionali

La divergenza di Kullback-Leibler (o entropia relativa) è centrale nell'ottimizzazione con vincoli di informazione.

Definizione

Per due distribuzioni p e q su lo stesso spazio:

$KL(p \parallel q) = \sum_i p_i \log(p_i / q_i)$ (caso discreto) $KL(p \parallel q) = \int p(x) \log(p(x) / q(x)) \, dx$ (caso continuo)

Interpretazione geometrica: proiezione informazionale

Il principio di massima entropia può essere reinterpretato come: trovare la distribuzione q più vicina (in senso di KL divergence) a una distribuzione di riferimento p_0 (spesso uniforme), soggetta a vincoli.

Problema di proiezione:

minimizza $KL(q \parallel p_0)$ soggetto a $E_{q[f_k]} = c_k$ per $k = 1, \dots, m$ $q \in \Delta_n$ (simplex delle probabilità)

Questo è un problema di ottimizzazione convessa (KL è convessa in q).

Teorema di Csiszár: La proiezione informazionale di p_0 sull'insieme dei vincoli esiste, è unica, e ha forma esponenziale come la MaxEnt.

Algoritmi iterativi: Iterative Scaling

Per trovare la distribuzione MaxEnt, l'algoritmo di Iterative Proportional Fitting (IPF) itera:

Inizializza $q^{(0)} = p_0$ Per $t = 0, 1, 2, \dots$: Per $k = 1, \dots, m$: $q_i^{(t+1)} = q_i^{(t)} \times (c_k / E_{q^{(t)}}[f_k])^{\alpha_k(i)}$ dove $\alpha_k(i)$ è un peso appropriato

Sotto condizioni di regolarità, IPF converge alla distribuzione MaxEnt.

Compressione Dati e Ottimizzazione

Codifica di Huffman come ottimizzazione

Il problema della codifica ottimale può essere formulato come problema di ottimizzazione.

Dato: Distribuzione $p_1, \dots, p_n$ di simboli Obiettivo: Assegnare codici binari di lunghezze $l_1, \dots, l_n$ minimizzando la lunghezza media:

$$L = \sum_{i=1}^n p_i l_i$$

soggetto alla disuguaglianza di Kraft: $\sum_{i=1}^n 2^{-l_i} \leq 1$

La disuguaglianza di Kraft è necessaria e sufficiente perché esista un codice binario prefisso con quelle lunghezze.

Soluzione via moltiplicatori di Lagrange

$$\text{Lagrangiana: } L(l, \lambda) = \sum_i p_i l_i + \lambda (\sum_i 2^{-l_i} - 1)$$

$$\text{Derivando rispetto a } l_i: \partial L / \partial l_i = p_i - \lambda \times 2^{-l_i} \times \log 2 = 0$$

$$\text{Quindi: } 2^{-l_i} = p_i / (\lambda \log 2)$$

$$\text{Sostituendo nella disuguaglianza di Kraft: } \sum_i p_i / (\lambda \log 2) = 1 \rightarrow \lambda = 1 / \log 2$$

$$\text{Quindi: } l_i = -\log_2 p_i$$

Poiché le lunghezze devono essere intere, arrotondiamo: $l_i = \lceil -\log_2 p_i \rceil$

La lunghezza media ottimale è:

$$H(p) / \log 2 \leq L < H(p) / \log 2 + 1$$

dove $H(p)$ è l'entropia di Shannon (in nats). Dividendo per $\log 2$ convertiamo in bit.

Questo stabilisce il teorema fondamentale di Shannon: l'entropia è il limite inferiore della compressione.

Codifica aritmetica e approssimazione dell'entropia

La codifica aritmetica elimina il gap di 1 bit della codifica di Huffman, avvicinandosi arbitrariamente all'entropia codificando sequenze di simboli invece di simboli singoli.

Informazione di Fisher e Cramér-Rao

L'informazione di Fisher connette statistica, geometria e ottimizzazione.

Definizione

Per una famiglia di distribuzioni parametrizzate $p(x | \theta)$, l'informazione di Fisher è:

$$I(\theta) = E[(\partial \log p(X | \theta) / \partial \theta)^2] = -E[\partial^2 \log p(X | \theta) / \partial \theta^2]$$

(le due forme sono equivalenti sotto regolarità).

Interpretazione geometrica

$I(\theta)$ misura la curvatura della log-likelihood. Alta informazione di Fisher significa che la likelihood cambia rapidamente con θ , quindi θ può essere stimato accuratamente dai dati.

Disuguaglianza di Cramér-Rao

Per ogni stimatore non distorto $\hat{\theta}$ di θ :

$$\text{Var}(\hat{\theta}) \geq 1 / I(\theta)$$

Questo è un limite inferiore fondamentale sulla varianza di qualsiasi stimatore. Stimatori che raggiungono questo limite (stimatori efficienti) sono ottimali.

Connessione con ottimizzazione: Natural Gradient

In ottimizzazione, il gradiente "naturale" usa la metrica dell'informazione di Fisher invece della metrica euclidea standard.

$$\text{Gradiente standard: } \theta_{t+1} = \theta_t - \eta \nabla_{\theta} L(\theta_t)$$

$$\text{Gradiente naturale: } \theta_{t+1} = \theta_t - \eta I(\theta_t)^{-1} \nabla_{\theta} L(\theta_t)$$

Il gradiente naturale è invariante per riparametrizzazioni e converge spesso più rapidamente, specialmente in problemi di ottimizzazione su varietà statistiche (come in deep learning con reti neurali profonde).

Free Energy Principle e Ottimizzazione Bayesiana

Il principio di energia libera, proposto dal neuroscienziato Karl Friston, formula percezione e azione come minimizzazione di energia libera.

Energia libera variazionale

Dato modello generativo $p(x, z | \theta)$ dove x sono dati osservati, z variabili latenti, θ parametri:

La log-verosimiglianza può essere decomposta:

$$\log p(x | \theta) = \text{ELBO}(q, \theta) + \text{KL}(q(z) || p(z | x, \theta))$$

dove:

- $\text{ELBO} = E_q[\log p(x, z | \theta)] - E_q[\log q(z)]$ (Evidence Lower Bound)
- $q(z)$ è una distribuzione variazionale approssimata

L'energia libera variazionale è definita come $F = -\text{ELBO}$:

$$F(q, \theta) = E_q[\log q(z)] - E_q[\log p(x, z | \theta)]$$

Minimizzare F rispetto a q equivale a massimizzare ELBO, che approssima la massimizzazione della log-verosimiglianza.

Algoritmo EM (Expectation-Maximization)

EM è un algoritmo di ottimizzazione per massimizzare la likelihood in presenza di variabili latenti, basato su minimizzazione alternata di F .

E-step: Fissa θ , minimizza F rispetto a $q \rightarrow q^*(z) = p(z | x, \theta)$ (distribuzione posteriore)

M-step: Fissa $q = q^*$, minimizza F rispetto a $\theta \rightarrow \theta^* = \operatorname{argmax}_{\theta} E_{q^*}[\log p(x, z | \theta)]$

Iterare fino a convergenza.

Teorema: EM aumenta monotonicamente la log-likelihood (non può peggiorare).

Dimostrazione:

Sia $L(\theta) = \log p(x | \theta)$. Dopo un ciclo EM completo, $\theta^{\text{old}} \rightarrow \theta^{\text{new}}$:

$L(\theta^{\text{new}}) = \text{ELBO}(q^*, \theta^{\text{new}}) + \text{KL}(q^* || p(z | x, \theta^{\text{new}})) \geq \text{ELBO}(q^*, \theta^{\text{new}})$ ($\text{KL} \geq 0$) $\geq \text{ELBO}(q^*, \theta^{\text{old}})$ (M-step massimizza ELBO) $= L(\theta^{\text{old}})$ (E-step rende $\text{KL} = 0$)

Reti Neurali e Information Bottleneck

Information Bottleneck Theory

Tishby e collaboratori hanno proposto di comprendere le reti neurali profonde attraverso il principio di information bottleneck.

Consideriamo un problema di apprendimento supervisionato: dato input X , predire output Y . Una rappresentazione intermedia $T = f(X)$ (es: attivazioni di un layer nascosto) deve:

1. Preservare informazione su Y : $I(T; Y)$ alta
2. Scartare informazione irrilevante su X : $I(T; X)$ bassa

Questo è un problema di ottimizzazione bi-obiettivo:

massimizza $I(T; Y)$ minimizza $I(T; X)$

Combinando con un trade-off parameter β :

massimizza $I(T; Y) - \beta I(T; X)$

Connessione con compressione e generalizzazione

Il trade-off cattura il dilemma bias-varianza:

- $I(T; X)$ bassa $\rightarrow T$ è una rappresentazione compressa, meno soggetta a overfitting
- $I(T; Y)$ alta $\rightarrow T$ preserva informazione predittiva

Reti che comprimono bene generalizzano bene.

Evidenza empirica

Studi hanno mostrato che il training di reti neurali ha due fasi:

1. **Fitting phase:** $I(T; Y)$ e $I(T; X)$ entrambe aumentano (rete memorizza training set)
2. **Compression phase:** $I(T; Y)$ rimane alta, $I(T; X)$ diminuisce (rete comprime, rimuove dettagli irrilevanti)

Questo suggerisce che la generalizzazione emerge dalla compressione.

Entropia Regolarizzata in Reinforcement Learning

Nel reinforcement learning, l'entropia della policy $\pi(a | s)$ (probabilità di azione a in stato s) è usata come regolarizzatore.

Maximum Entropy RL

Obiettivo standard: massimizza ritorno atteso $\sum_t \gamma^t r_t$

Obiettivo MaxEnt RL: massimizza $\sum_t \gamma^t [r_t + \alpha H(\pi(\cdot | s_t))]$

dove $\alpha > 0$ controlla il trade-off tra ricompensa ed esplorazione (alta entropia).

Vantaggi:

1. Promuove esplorazione (policy stocastiche, non deterministiche)
2. Robustezza (policy non collassa su singola azione)
3. Migliore trasferibilità (policy più generale)

Algoritmo SAC (Soft Actor-Critic)

SAC ottimizza l'obiettivo MaxEnt usando:

- Soft Bellman equation: $Q^*(s, a) = r(s, a) + \gamma E[V^*(s') + \alpha H(\pi(\cdot | s'))]$
- Policy update massimizzando $Q - \alpha \log \pi$ (trade-off reward-entropy)

SAC è uno degli algoritmi RL continui più efficaci, applicato a robotica, controllo, giochi.

13.8 Complessità di Kolmogorov e Compressione Ottimale

La complessità di Kolmogorov $K(x)$ di una stringa x è la lunghezza del programma più corto che genera x .

$$K(x) = \min\{|p| : U(p) = x\}$$

dove U è una macchina di Turing universale, p è un programma, $|p|$ è la sua lunghezza.

Teorema: $K(x)$ è incomputabile.

Dimostrazione (sketch via paradosso di Berry):

Supponiamo K computabile. Definiamo: "Il più piccolo intero n tale che $K(n) > 1000$ "

Questo intero può essere specificato con circa 100 bit (la descrizione sopra). Ma per definizione, $K(n) > 1000 \rightarrow$ contraddizione.

Relazione con entropia

Per una variabile casuale X :

$$E[K(X)] \approx H(X) + O(1)$$

Questa equivalenza asintotica connette complessità algoritmica (teoria della computazione) ed entropia (teoria dell'informazione).

Minimum Description Length (MDL)

Principio di selezione di modelli: tra modelli che spiegano i dati, scegli quello con descrizione più corta.

Per modello M e dati D :

$$\text{Descrizione totale} = \text{Lunghezza}(M) + \text{Lunghezza}(D | M)$$

dove $\text{Lunghezza}(D | M)$ è la lunghezza per codificare i dati dato il modello.

MDL bilancia complessità del modello (primo termine) e qualità del fit (secondo termine), prevenendo overfitting.

MDL è equivalente (asintoticamente) a criteri Bayesiani come BIC (Bayesian Information Criterion).

Ottimizzazione e Termodinamica: Connessioni Profonde

Simulated Annealing e meccanica statistica

Simulated Annealing è direttamente ispirato dalla termodinamica.

In meccanica statistica, la distribuzione di Boltzmann è:

$$p(s) \propto \exp(-E(s) / (k_B T))$$

dove $E(s)$ è energia dello stato s , T temperatura, k_B costante di Boltzmann.

In SA, $E(s)$ è la funzione obiettivo, T è temperatura dell'algoritmo:

$$P(\text{accetta stato peggiore}) = \exp(-\Delta E / T)$$

Teorema: SA con schedule di raffreddamento logaritmico $T(t) = T_0 / \log(t)$ converge alla distribuzione di Boltzmann concentrata sugli ottimi globali.

Entropia termodinamica e Shannon

L'entropia termodinamica S e l'entropia di Shannon H sono profondamente connesse.

Teorema (Shannon, Jaynes): L'entropia termodinamica è uguale (a meno di costante) all'entropia di Shannon della distribuzione microstati:

$$S = k_B \times H(p)$$

dove p è la distribuzione di probabilità su microstati.

Questo unifica termodinamica e teoria dell'informazione: l'entropia misura sia disordine fisico che incertezza informazionale.

Conclusioni: Ottimizzazione come Inferenza Informazionale

La connessione tra ottimizzazione e teoria dell'informazione è profonda e multiforme:

1. **Principio MaxEnt:** Ottimizzazione vincolata fornisce distribuzioni razionali data informazione parziale
2. **Compressione:** Codici ottimali minimizzano lunghezza media, limite dato da entropia
3. **Apprendimento:** Minimizzare energia libera variazionale è simultaneamente inferenza bayesiana e ottimizzazione
4. **Generalizzazione:** Compressione informazionale (information bottleneck) spiega perché reti neurali generalizzano
5. **Esplorazione:** Massimizzare entropia bilanciando ricompensa produce policy robuste in RL

Questa unificazione suggerisce che ottimizzazione, inferenza probabilistica, e processamento dell'informazione sono aspetti diversi di una teoria più profonda. Il framework moderno dell'apprendimento automatico sintetizza questi elementi: modelli generativi (probabilità), ottimizzazione (gradiente), e informazione (entropia, complessità) convergono in algoritmi come VAE, GAN, e trasformatori.

La frontiera attuale esplora come questi principi si estendono a contesti più generali: geometria dell'informazione, ottimizzazione su varietà, apprendimento causale, e teoria dei giochi algoritmici, tutti unificati dalla visione di ottimizzazione come processamento ottimale dell'informazione sotto vincoli computazionali e informativi.

OTTIMIZZAZIONE STOCASTICA

Metodi Stocastici per l'Ottimizzazione

Gradient Descent Stocastico: Fondamenti

Il gradient descent stocastico (SGD) è diventato il metodo dominante per l'ottimizzazione in machine learning, specialmente per reti neurali profonde. La sua potenza deriva dalla capacità di gestire dataset enormi senza richiedere il calcolo del gradiente esatto.

Il problema dell'ottimizzazione empirical risk

In machine learning supervisionato, vogliamo minimizzare il rischio empirico:

$$f(\theta) = (1/N) \sum_{i=1}^N L(\theta; x_i, y_i)$$

dove $L(\theta; x, y)$ è la loss per il singolo esempio (x, y) , e θ sono i parametri del modello.

Batch Gradient Descent

L'approccio classico calcola il gradiente completo:

$$\theta_{t+1} = \theta_t - \eta_t \nabla f(\theta_t) = \theta_t - (\eta_t/N) \sum_{i=1}^N \nabla L(\theta_t; x_i, y_i)$$

Problema: Per N grande (milioni o miliardi di esempi), calcolare la somma completa ad ogni iterazione è proibitivo.

Stochastic Gradient Descent

SGD seleziona casualmente un singolo esempio i_t a ogni iterazione:

$$\theta_{t+1} = \theta_t - \eta_t \nabla L(\theta_t; x_{i_t}, y_{i_t})$$

Osservazione chiave: Questo è un estimatore non distorto del gradiente vero:

$$E[\nabla L(\theta_t; x_{i_t}, y_{i_t})] = (1/N) \sum_i \nabla L(\theta_t; x_i, y_i) = \nabla f(\theta_t)$$

Mini-batch SGD

Il compromesso pratico usa mini-batch di dimensione $B \ll N$:

$$\theta_{t+1} = \theta_t - (\eta_t/B) \sum_{i \in B_t} \nabla L(\theta_t; x_i, y_i)$$

dove B_t è un batch casuale di dimensione B .

Vantaggi:

- Più stabile di SGD puro (varianza ridotta di fattore B)
- Sfrutta parallelismo hardware (GPU calcola gradienti su batch in parallelo)
- B tipico: 32, 64, 128, 256

Analisi di Convergenza di SGD

Teorema (Convergenza di SGD per funzioni convesse)

Sia $f: \mathbb{R}^d \rightarrow \mathbb{R}$ convessa e L -lipschitz. Sia ∇_t un estimatore stocastico del gradiente con:

$$E[\|\nabla_{\theta_t}\|] = \nabla f(\theta_t) \quad E[\|\nabla_{\theta_t} - \nabla f(\theta_t)\|^2] \leq \sigma^2$$

Se usiamo learning rate $\eta_t = \eta_0/\sqrt{t}$, allora:

$$E[f(\bar{\theta}_T)] - f^* \leq O(1/\sqrt{T})$$

dove $\bar{\theta}_T = (1/T) \sum_{t=1}^T \theta_t$ è la media delle iterate e f^* è il minimo.

Sketch della dimostrazione

Consideriamo la distanza al minimo θ^* :

$$\|\theta_{t+1} - \theta^*\|^2 = \|\theta_t - \eta_t \nabla_{\theta_t} - \theta^*\|^2 = \|\theta_t - \theta^*\|^2 - 2\eta_t \nabla_{\theta_t} \cdot (\theta_t - \theta^*) + \eta_t^2 \|\nabla_{\theta_t}\|^2$$

Prendendo valore atteso e usando convessità ($f(\theta_t) - f^* \leq \nabla f(\theta_t) \cdot (\theta_t - \theta^*)$):

$$E[\|\theta_{t+1} - \theta^*\|^2] \leq E[\|\theta_t - \theta^*\|^2] - 2\eta_t (f(\theta_t) - f^*) + \eta_t^2 (E[\|\nabla f(\theta_t)\|^2] + \sigma^2)$$

Usando bound sui gradienti e sommando su t , si ottiene il risultato.

Caso non-convesso

Per funzioni non-convesse (come reti neurali), la convergenza a minimo globale non è garantita. Tuttavia, possiamo dimostrare convergenza a punti stazionari ($\nabla f = 0$).

Teorema: Per f smooth (∇f L -lipschitz) e lower-bounded, con $\eta_t = \eta_0/\sqrt{t}$:

$$(1/T) \sum_{t=1}^T E[\|\nabla f(\theta_t)\|^2] \leq O(1/\sqrt{T})$$

Questo garantisce che in media i gradienti diventano piccoli $\rightarrow$ convergenza a punto critico.

Varianza e Metodi di Riduzione della Varianza

Il problema principale di SGD è la varianza: il gradiente stocastico è rumoroso. Questo rallenta la convergenza specialmente vicino all'ottimo.

SVRG (Stochastic Variance Reduced Gradient)

SVRG, proposto da Johnson e Zhang (2013), riduce la varianza mantenendo costo computazionale paragonabile a SGD.

Algoritmo SVRG:

Inizializza $\tilde{\theta}_0$

Per epoch $s = 1, 2, \dots$:

$$\theta_0^s = \tilde{\theta}_{s-1}$$

$$\mu = (1/N) \sum_{i=1}^N \nabla L(\theta_0^s; x_i, y_i) \quad \# \text{ Full gradient}$$

Per $t = 1$ a m :

Scegli i_t uniformemente da $\{1, \dots, N\}$

$$g_t = \nabla L(\theta_t^s; x_{\{i_t\}}, y_{\{i_t\}}) - \nabla L(\theta_0^s; x_{\{i_t\}}, y_{\{i_t\}}) + \mu$$

$$\theta_{\{t+1\}}^s = \theta_t^s - \eta g_t$$

$$\tilde{\theta}_s = \theta_m^s \text{ (oppure casualmente da } \{\theta_t^s\})$$

Perché funziona:

L'estimatore del gradiente è:

$$g_t = \nabla L(\theta_t; x_i) - \nabla L(\theta_0; x_i) + \mu$$

dove $\mu = \nabla f(\theta_0)$ è il gradiente completo al checkpoint θ_0 .

Proprietà:

1. $E[g_t] = \nabla f(\theta_t)$ (non distorto)
2. $\text{Var}[g_t] \rightarrow 0$ quando $\theta_t \rightarrow \theta^*$ (varianza decade)

Vicino all'ottimo, $\nabla L(\theta_t; x_i) \approx \nabla L(\theta_0; x_i)$, quindi il primo termine si cancella, e $g_t \approx \mu$ che ha varianza zero.

Convergenza: SVRG raggiunge tasso lineare $O(e^{-T})$ per funzioni fortemente convesse, mentre SGD standard raggiunge solo $O(1/T)$.

SAGA

SAGA (Defazio et al., 2014) è simile a SVRG ma mantiene una tabella di tutti i gradienti individuali:

$$g_i^t = \nabla L(\theta; x_i, y_i) \text{ calcolato all'ultima volta che } x_i \text{ fu visitato}$$

Update:

$$\theta_{\{t+1\}} = \theta_t - \eta [\nabla L(\theta_t; x_{\{i_t\}}, y_{\{i_t\}}) - g_{\{i_t\}}^t + (1/N) \sum_i g_i^t]$$

Quando visitiamo $x_{\{i_t\}}$, aggiorniamo: $g_{\{i_t\}}^{t+1} = \nabla L(\theta_{\{t+1\}}; x_{\{i_t\}}, y_{\{i_t\}})$

Costo memoria: $O(Nd)$ per memorizzare tutti i gradienti, ma in pratica fattibile e molto efficace.

Momentum e Accelerazione

I metodi di momentum migliorano SGD accumulando una media mobile dei gradienti passati.

Classical Momentum (Heavy Ball)

$$\theta_{t+1} = \theta_t - \eta \nabla f(\theta_t) + \beta(\theta_t - \theta_{t-1})$$

Equivalentemente, con velocità $v_t = \theta_t - \theta_{t-1}$:

$$v_{t+1} = \beta v_t - \eta \nabla f(\theta_t) \quad \theta_{t+1} = \theta_t + v_{t+1}$$

Parametro tipico: $\beta = 0.9$

Interpretazione fisica: La particella ha inerzia. I gradienti applicano forze che cambiano la velocità, non direttamente la posizione.

Nesterov Accelerated Gradient (NAG)

NAG "guarda avanti" prima di calcolare il gradiente:

$$v_{t+1} = \beta v_t - \eta \nabla f(\theta_t + \beta v_t) \quad \theta_{t+1} = \theta_t + v_{t+1}$$

Questo anticipa dove sarà il prossimo punto e calcola il gradiente lì, risultando in convergenza più rapida.

Teorema (Nesterov, 1983): Per funzioni convesse smooth, NAG converge a tasso $O(1/t^2)$ invece di $O(1/t)$ del gradient descent standard.

Questa accelerazione è ottimale tra tutti i metodi del primo ordine (che usano solo gradienti, non hessiane).

Momentum in versione stocastica

Combining SGD con momentum:

$$v_{t+1} = \beta v_t - \eta \nabla_t \theta_{t+1} \quad \theta_{t+1} = \theta_t + v_{t+1}$$

dove ∇_t è il gradiente stocastico (mini-batch).

In pratica, momentum stabilizza SGD e accelera convergenza, specialmente in paesaggi con ravines (valli strette).

Metodi Adattivi: AdaGrad, RMSProp, Adam

I metodi adattivi modificano il learning rate per ciascun parametro in base alla storia dei gradienti.

AdaGrad (Adaptive Gradient)

Idea: Parametri con gradienti grandi (frequentemente aggiornati) ricevono learning rate ridotto. Parametri con gradienti piccoli (raramente aggiornati) mantengono learning rate alto.

Algoritmo:

Inizializza $G_0 = 0$ (matrice diagonale $d \times d$) Per $t = 1, 2, \dots$: $g_t = \nabla_t$ (gradiente stocastico)
 $G_t = G_{t-1} + \text{diag}(g_t \odot g_t)$ ($\odot$ = prodotto elemento per elemento) $\theta_{t+1} = \theta_t - (\eta / \sqrt{G_t + \epsilon}) \odot g_t$

dove la divisione e radice sono elemento per elemento, $\epsilon > 0$ è un piccolo termine di stabilizzazione (tipicamente 10^{-8}).

Proprietà:

- Parametro i con $\sum_{\tau} g_{\tau,i}^2$ grande $\rightarrow$ learning rate piccolo
- Automaticamente adatta scale dei parametri
- Ottimo per gradienti sparsi (es: embedding in NLP)

Problema: G_t cresce monotonamente $\rightarrow$ learning rate decade troppo rapidamente $\rightarrow$ l'algoritmo può fermarsi prematuramente.

RMSProp (Root Mean Square Propagation)

RMSProp risolve il problema di AdaGrad usando media mobile esponenziale invece di somma cumulativa:

$$v_t = \gamma v_{t-1} + (1 - \gamma) g_t^2 \text{ (elemento per elemento)} \quad \theta_{t+1} = \theta_t - (\eta / \sqrt{v_t + \epsilon}) \odot g_t$$

Parametro tipico: $\gamma = 0.9$

Ora v_t è una stima della varianza recente del gradiente, e il learning rate si adatta dinamicamente ma non decade monotonamente.

Adam (Adaptive Moment Estimation)

Adam combina momentum (primo momento) con scaling adattivo (secondo momento):

$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$ (primo momento, come momentum) $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ (secondo momento, come RMSProp)

Bias correction (importante nelle prime iterazioni):

$$\hat{m}_t = m_t / (1 - \beta_1^t) \quad \hat{v}_t = v_t / (1 - \beta_2^t)$$

Update:

$$\theta_{t+1} = \theta_t - \eta \times \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$$

Parametri standard: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$

Perché la bias correction:

All'inizio, $m_0 = v_0 = 0$. Dopo una iterazione:

$$m_1 = (1 - \beta_1)g_1 \rightarrow \text{molto più piccolo di } g_1 \text{ se } \beta_1 \approx 1$$

Correggendo: $\hat{m}_1 = m_1 / (1 - \beta_1) = g_1$, che è non distorto.

Adam è diventato lo standard de facto per training di reti neurali profonde grazie a:

- Convergenza rapida
- Robusto a scelta di hyperparametri
- Funziona bene su vasta gamma di problemi

Varianti di Adam

- **AdamW**: Corregge weight decay (regolarizzazione L2) in Adam
- **RAdam**: Rectified Adam, corregge fase iniziale di warmup
- **AdaBound**: Combina Adam con SGD, passando dinamicamente da uno all'altro

Ottimizzazione Stocastica Vincolata

Molti problemi richiedono vincoli: $\theta \in C$ dove C è un insieme convesso.

Projected Gradient Descent

$$\theta_{t+1} = \Pi_C[\theta_t - \eta_t \nabla f_t]$$

dove $\Pi_C[\cdot]$ è la proiezione euclidea su C :

$$\Pi_C[\theta] = \operatorname{argmin}_{\theta' \in C} \|\theta - \theta'\|^2$$

Esempi di proiezioni

1. **Simplex** ($\sum_i \theta_i = 1, \theta_i \geq 0$): Algoritmo efficiente $O(d \log d)$ di Duchi et al.
2. **Box constraints** ($a_i \leq \theta_i \leq b_i$): $\Pi_C[\theta]_i = \max(a_i, \min(\theta_i, b_i))$
3. **Norma limitata** ($\|\theta\| \leq R$): $\Pi_C[\theta] = \theta$ se $\|\theta\| \leq R$, altrimenti $(R/\|\theta\|)\theta$

Mirror Descent

Generalizzazione che usa proiezioni Bregman invece di euclidee.

Definiamo divergenza di Bregman per funzione convessa φ :

$$D_\varphi(\theta, \theta') = \varphi(\theta) - \varphi(\theta') - \nabla \varphi(\theta') \cdot (\theta - \theta')$$

Mirror descent:

$$\theta_{t+1} = \operatorname{argmin}_{\theta \in C} \{\eta_t g_t \cdot \theta + D_\varphi(\theta, \theta_t)\}$$

Per $\varphi(\theta) = (1/2)\|\theta\|^2$, recuperiamo projected gradient descent.

Per $\varphi(\theta) = \sum_i \theta_i \log \theta_i$ (entropia negativa), otteniamo l'algoritmo di multiplicative weights, utile per ottimizzazione su simplex.

Stochastic Second-Order Methods

I metodi del secondo ordine usano informazione sulla curvatura (Hessiano).

Problema: Calcolare e invertire l'Hessiano $H \in \mathbb{R}^{d \times d}$ costa $O(d^3)$, proibitivo per d grande (milioni in deep learning).

L-BFGS (Limited-memory BFGS)

Approssima l'Hessiano inverso usando solo ultimi m vettori di gradiente/posizione:

$$s_i = \theta_{i+1} - \theta_i \quad y_i = \nabla f(\theta_{i+1}) - \nabla f(\theta_i)$$

L'approssimazione dell'Hessiano inverso H_t^{-1} è costruita ricorsivamente senza memorizzare matrice completa.

Costo: $O(md)$ per iterazione con $m \approx 10-20$ tipicamente.

L-BFGS stocastico

Estensioni stocastiche di L-BFGS esistono ma sono delicate:

- Problemi con rumore del gradiente stocastico
- Difficile stimare curvatura accuratamente con mini-batch

Approcci recenti:

- **Stochastic L-BFGS:** Usa batch più grandi per curvatura
- **AdaHessian:** Approssima diagonale dell'Hessiano per scaling adattivo

Coordinate Descent Stocastico

Per problemi con molte variabili, possiamo aggiornare una coordinata (o blocco) alla volta.

Randomized Coordinate Descent

Scegli coordinata i_t uniformemente $\theta_{t+1} = \theta_t \quad \theta_{t+1}^{i_t} = \theta_t^{i_t} - \eta_t \nabla_{i_t} f(\theta_t)$

Teorema: Per f convessa smooth separabile, coordinate descent converge a tasso $O(1/t)$, uguale a gradient descent ma aggiornando solo $1/d$ delle variabili per iterazione.

Quando è vantaggioso:

- Calcolare gradiente parziale ∇_{i_t} è molto più economico che ∇f completo
- Sparse data (solo poche coordinate sono non-zero)
- Problemi strutturati (es: LASSO, SVM)

Dual Coordinate Ascent

Per problemi con duale più trattabile del primale, coordinate descent sul duale può essere molto efficiente.

Esempio: Training di SVM. Il duale è:

$$\max_{\alpha} \sum_i \alpha_i - (1/2) \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \text{ soggetto a } 0 \leq \alpha_i \leq C, \sum_i \alpha_i y_i = 0$$

Coordinate ascent su α converge rapidamente, specialmente con selezione euristica delle coordinate.

Zeroth-Order (Derivative-Free) Stochastic Optimization

Quando i gradienti non sono disponibili o troppo costosi, possiamo stimarli da valutazioni della funzione.

Finite Differences Stocastiche

Approssima gradiente usando perturbazioni:

$$\nabla f(\theta) = [f(\theta + \delta e_i) - f(\theta - \delta e_i)] / (2\delta) \text{ per coordinata } i$$

dove e_i è il vettore unitario. Richiede $2d$ valutazioni.

Smoothing Gaussiano

Approssima f con versione smoothed:

$$f_{\sigma}(\theta) = E_{\{u \sim N(0, I)\}}[f(\theta + \sigma u)]$$

Il gradiente di f_{σ} è:

$$\nabla f_{\sigma}(\theta) = E_u[(f(\theta + \sigma u)/\sigma) \times u]$$

Possiamo stimare stocasticamente:

$$\nabla f(\theta) = (1/(m\sigma)) \sum_{j=1}^m f(\theta + \sigma u_j) u_j$$

dove $u_j \sim N(0, I)$. Richiede solo m valutazioni (tipicamente $m \ll d$).

Applicazioni:

- Ottimizzazione di sistemi black-box (simulatori complessi)
- Reinforcement learning (policy gradient senza backpropagation)
- Hyperparameter tuning

Evolutionary Strategies come Zero-Order Optimization

ES può essere interpretato come gradient ascent con gradiente stimato da perturbazioni:

$$\theta_{t+1} = \theta_t + \alpha \times (1/(m\sigma^2)) \sum_{j=1}^m f(\theta_t + \sigma \varepsilon_j) \varepsilon_j$$

dove $\varepsilon_j \sim N(0, I)$. Questo è equivalente a natural gradient ascent su distribuzione gaussiana.

Convergence Diagnostics e Adaptive Learning Rate Scheduling

Learning Rate Scheduling

Strategie comuni per decadere il learning rate:

1. **Step decay:** $\eta_t = \eta_0 \times \gamma^{\lfloor t/T \rfloor}$ Riduce η di fattore γ ogni T epoche
2. **Exponential decay:** $\eta_t = \eta_0 \times e^{-\lambda t}$
3. **Polynomial decay:** $\eta_t = \eta_0 / (1 + t)^p$ con $p \in (0.5, 1]$
4. **Cosine annealing:** $\eta_t = \eta_{\min} + (\eta_0 - \eta_{\min})/2 \times [1 + \cos(\pi t/T)]$
5. **Warm restarts:** Periodicamente resetta η a η_0 , simula riavvii

Learning Rate Warmup

Per reti molto profonde, iniziare con η piccolo e aumentare gradualmente nelle prime epoche stabilizza training:

$$\eta_t = (t/T_{\text{warmup}}) \times \eta_0 \text{ per } t \leq T_{\text{warmup}}$$

Poi applica decay standard.

Adaptive Scheduling con Validation Loss

Monitora loss su validation set. Quando plateau (nessun miglioramento per k epoche), riduci η :

$$\eta_{\text{new}} = \eta_{\text{old}} \times \rho$$

con $\rho \approx 0.1-0.5$.

Questo è usato nella pratica per evitare decay troppo precoce o troppo lento.

Convergence Diagnostics

1. **Gradient norm:** $\|\nabla f(\theta_t)\| \rightarrow 0$ indica avvicinamento a punto critico
2. **Parameter updates:** $\|\theta_{t+1} - \theta_t\| \rightarrow 0$ indica convergenza
3. **Train/validation gap:** Divergenza crescente indica overfitting

4. **Loss oscillation:** Oscillazioni eccessive indicano η troppo alto

Conclusioni: Il Paesaggio Moderno dell'Ottimizzazione Stocastica

L'ottimizzazione stocastica è diventata la spina dorsale del machine learning moderno. I progressi chiave degli ultimi 15 anni includono:

Riduzione della varianza: SVRG, SAGA permettono convergenza lineare anche in setting stocastico, avvicinando le garanzie teoriche dell'ottimizzazione deterministica.

Metodi adattivi: Adam e varianti hanno democratizzato il deep learning, riducendo l'expertise necessaria per hyperparameter tuning e rendendo possibile il training di architetture sempre più profonde.

Comprensione teorica: Nonostante il successo empirico, la teoria sta ancora cercando di spiegare fenomeni come:

- Perché SGD generalizza meglio di metodi del secondo ordine?
- Quale ruolo gioca il rumore stocastico nell'sfuggire a minimi locali pessimi?
- Come interagiscono architettura, inizializzazione, e dinamica di ottimizzazione?

Frontiere attuali:

1. **Large-batch training:** Come scalare a batch enormi (migliaia) mantenendo convergenza?
2. **Federated learning:** Ottimizzazione distribuita con dati decentralizzati e comunicazione limitata
3. **One-shot e few-shot optimization:** Raggiungere buone soluzioni con poche iterazioni per deployment rapido
4. **Neural architecture search:** Ottimizzare simultaneamente architettura e parametri
5. **Adversarial robustness:** Ottimizzare contro perturbazioni avversarie

L'ottimizzazione stocastica continua ad evolversi, spinta dalle esigenze pratiche del machine learning su scala sempre crescente e da domande teoriche profonde sulla natura dell'apprendimento e della generalizzazione.